МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ГОУ ВПО «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ УПРАВЛЕНИЯ И ПРЕДПРИНИМАТЕЛЬСТВА

КАФЕДРА ИНФОРМАЦИОННОГО МЕНЕДЖМЕНТА

Ломакин В.В.

БАЗЫ ДАННЫХ И БАЗЫ ЗНАНИЙ

Учебное пособие

Белгород 2010г.

УДК 004.65(07)
Л74

Печатается по решению
редакционно-издательского совета
Белгородского государственного университета

Рецензенты:
Синюк В.Г., к.т.н., профессор кафедры программного обеспечения вычислительной техники и автоматизированных систем Белгородского государственного технологического университета им. В.Г. Шухова
Михелев В.М., к.т.н., доцент кафедры математического и программного обеспечения информационных систем Белгородского государственного университета

Л74 Ломакин В.В.
Базы данных и базы знаний: Учебное пособие / Сост. Ломакин В.В. – Белгород: Изд-во БелГУ, 2010. – 216с.

Учебное пособие по курсу «Базы данных и базы знаний» для студентов высших учебных заведений содержит теоретические аспекты построения баз данных и практические подходы к реализации баз данных в СУБД Access. Изложены особенности организации баз знаний, способы представления знаний в интеллектуальных системах, ключевые аспекты разработки систем, основанных на знаниях. Учебное пособие составлено в соответствии с требованиями к обязательному минимуму содержания и уровню подготовки специалиста с высшим образованием согласно Государственному образовательному стандарту высшего профессионального образования по специальности 080508.65 «Информационный менеджмент».

УДК 004.65 (07)

Предисловие

«Базы данных и базы знаний» – специальная дисциплина, рассматривающая теоретические и практические аспекты построения баз данных, организации баз знаний в контексте решения задач управления информационными системами на предприятиях различной топологии.

Материал курса разработан на основе анализа современного состояния вопросов организации управления на предприятии, связанных с организацией и методами построения информационных систем. В современных условиях решающая роль отводится возможностям гибкости, мобильности и адаптации систем управления предприятием к изменяющимся условиям рыночной экономики, что достигается использованием современных средств проектирования и реализации информационного обеспечения, основу которых составляют базы данных и базы знаний.

Представленный в учебном пособии материал ориентирован на сочетание изучения теоретических положений, касающихся состава, структуры, методов формального построения информационной структуры управления предприятием, с освоением практических методов проектирования и реализации задач, решаемых фрагментами предметной области деятельности предприятия. Особое место отводится индивидуальной работе студентов по изучению и моделированию предметной области конкретного предприятия (подразделения) с целью получения навыков самостоятельной выработки подходов к структурному построению концептуальной модели предметной области, использованию инструментальных средств реализации, а также проектированию и реализации информационной системы для решения локальных задач управления предприятием.

Важной особенностью учебного пособия является изложение практических аспектов построения баз данных с помощью популярной СУБД Access. Рассмотрена последовательность разработки предметно-ориентированных БД, представлены сведения о типах полей таблиц БД, организации связей между таблицами, разработке форм и отчетов, составлении запросов.

Курс «Базы данных и базы знаний» имеет связь с такими общепрофессиональными и специальным дисциплинами, как «Программирование и программное обеспечение информационных технологий», «Информатика», «Технические средства информационных технологий, вычислительные машины, сети и системы телекоммуникаций».

Тема 1. Теоретические основы баз данных

Цели и задачи изучения темы

Целью изучения темы является формирование устойчивого представления об основах проектирования и разработки баз данных информационных систем, применения систем управления базами данных.

Задачи изучения темы:

- изучить существующие структуры данных;
- изучить основные этапы эволюции технологий обработки данных;
- рассмотреть существующие модели организации данных в СУБД;
- изучить понятие и виды нормализации отношений при проектировании реляционных БД;
- рассмотреть архитектуру информационных систем;
- познакомиться с основами работы в СУБД Access;
- рассмотреть методы разработки баз данных.

1.1. Структуры данных

Значимость информационных ресурсов определяет то, что на их основе осуществляется управление другими ресурсами. Чтобы использование информационных ресурсов было эффективным, требуется разработка соответствующей технологии. Для обработки данных с помощью ЭВМ данные нужно поместить в память с учетом способа внутреннего представления данных.

Описание структуры данных при решении прикладных задач осуществляется с помощью универсальных языков программирования или языков-спецификаций, которые представляют объект лишь с определенной уровнем адекватности.

Так как реальных объектов большое количество для их описания используют универсальные структуры данных, которые классифицируются так:

1. *Простые* типы данных не обладают внутренней структурой и представляют собой конечный набор базисных типов:
   - логический;
   - целый;
   - вещественный;
   - перечисляемый (для перечисления значений, входящих в тип);
   - интервальный;
   - символьный (используется для образования текста).
2. *Структурированные* типы данных предназначены для конструирования из конечного набора базисных типов более сложных структур, основными из которых являются:
   - функция с конечной областью определения – отображение конечного множества данных одного типа на множество данных другого типа, называемая в большинстве языков программирования «массив»;
   - декартово произведение – компоненты, которые противоположны с массивом, могут иметь разные типы данных (записи, таблицы);
   - объединение – множество, отдельные элементы которого классифицируются по категориям;
   - последовательность – конечное, но не фиксированное число элементов;
   - рекурсивные структуры предназначены для более сложных преобразований.

С точки зрения обработки данных интересны последовательности, разновидностями которых являются файлы последовательного и прямого доступа.

3. *Файл* – логическое понятие, относящееся в основном к организации данных, его употребляют, когда конкретный физический носитель либо не интересен, либо всегда одинаков.

Выделяют класс динамических структур, которые могут изменять свой размер в зависимости от этапа выполнения программы.

4. *Стек* – динамическая структура данных, основанная на принципе LIFO (Last In – First Out). По аналогии с магазином автомата стек так же называют стек.

5. *Очередь* – это FIFO(First In – Last Out).

6. *Линейный список* предназначен для создания сложных динамических структур, организованных с помощью ссылок в определенном порядке. Линейный список удобно модифицировать только с помощью указателей, то есть, меняя ссылки. Наиболее эффективно линейный список организуется с помощью *ссылочного* типа данных.

Ссылочные типы данных предназначены для обеспечения ссылок на другие типы данных и называются *указателями*. Они принимаются для процедурных языков программирования (Паскаль и С++), где существуют понятия *области данных для их хранения*. На базе этого типа просто и эффективно реализовать динамические структуры данных, так как стек, который основан на принципе «Последним пришел – первым вышел» - Last In – First Out (LIFO) и очередь: «First In – Last Out», которая в отличие от стека имеет две точки: вход и выход.

Программы для решения определенных задач можно рассматривать как объединение структур данных и алгоритма:

Алгоритм + Структура данных = Программа

На основе применения структур данных совершается переход от реальных объектов к некоторому уровню их абстракции.

1.2. Эволюция технологии обработки данных

Внедрение ЭВМ позволило перейти к автоматизации обработки данных. На первом этапе использования ЭВМ данные накапливали и хранили в виде отдельных данных, по мере возникновения новых потребностей создавались новые файлы.

Прикладные программы разрабатывались с учетом имеющихся файлов данных, при этом много информации неявным образом содержались в самих программах обработки файлов. Такая структура хранения имеет следующие недостатки:

- требуемую информацию трудно было получить, т.к. необходимо знание языков программирования и организации файлов;
- качество принимаемых решений было неудовлетворительным из-за отсутствия целостности данных. Ответы на идентичные запросы могут быть разными из-за дублирования информации и непоследовательности ее обновления, что было следствием разобщенности файлов.

Развитие технических средств и средств программирования сделало возможным накопление и обработку больших объемов данных. В настоящее время определяющим направлением организации и обработки файлов стала концепция баз данных.

База данных (БД) – это совокупность взаимосвязанных данных, используемых группой пользователей и хранящаяся с регулируемой избыточностью. Хранимые данные не зависят от программ пользователя. Это основное положение концепции БД. Именно то, что теперь информация о структуре данных хранится

непосредственно в файлах данных, при этом формат файла стандартизирован.

Широкое использование БД в управлении предприятием обусловлено оперативностью, гибкостью (получение ответов на любые запросы, наличие эффективных методов модификации данных и внесения изменений), безопасностью, целостностью, доступностью (возможность использования всей совокупности данных в БД для каждого пользователя).

В конце 1960-х гг. появилась идея создания интегрированной базы предприятия, но она оказалась практически недостижимой из-за сложностей проектирования и сопровождения. Эта причина, а так же появление персональных ЭВМ, развитие локальных вычислительных сетей способствовали появлению распределенных систем.

Процесс проектирования и функционирования распределенных БД требует централизованного планирования и управления. Можно выделить следующие способы распределения данных:

- одинаковые копии данных хранятся в разных местах использования, так как это дешевле передачи данных, модификация данных контролируется централизованно – *копируемые данные*;
- группы данных совместимые с исходной базой данных хранятся отдельно для местной обработки – это *подмножество данных*;
- данные в системе интегрируются при передаче на более высокий уровень управления – *реорганизованные данные*;
- на различных объектах используются одинаковые структуры, но хранятся разные данные – *секционированные данные*;
- на различных объектах используются различные структуры данных, объединяемые в интегрируемые системы – *данные с отдельной схемой*;
- независимые БД подразделений, спроектированные без координации, требующие объединения – *несовместимые данные*.

Основная предпосылка современного подхода к обработке информации – относительная стабильность данных. Это означает, что значения данных сменяются постоянно, а их структура остается стабильной.

Процедуры обработки данных, для которых структура меняется быстро, оказываются менее жизнеспособными, чем те, которые основаны на относительной стабильности.

Источниками данных в любой ИС являются объекты, их свойства, процессы, функции, выполняемые над объектами или для них.

При проектировании БД предметная область рассматривается в виде реального представления объекта, его описания в формальном виде и данных, которые отражают это представление (информационное представление). Информационная модель считается базовым понятием информатики. СУБД выполняет роль специально обобщенного инструментария для использования данных.

1.3. Характеристики СУБД на базе персональных ЭВМ и элементы проектирования ИС

СУБД является типовым элементом при проектировании автоматизированных систем, поэтому ее правильный выбор во многом определяет эффективность системы. Программные составляющие СУБД включают ядро и сервисные средства.

Ядро – это набор модулей, необходимых и достаточных для содержания и поддержания БД.

Сервисные средства – это дополнительные возможности и услуги по обслуживанию ИС.

Наиболее существенными характеристиками для составления СУБД являются:

1. Операционная среда, то есть модель ЭВМ, версия ОС, требования к конфигурации комплекса технических средств для реализации системы.
2. Область использования.
3. Логическая и физическая организация данных.
4. Целостность, безопасность, расширяемость.
5. Мощность языковых средств.
6. Возможности сервисных средств.
7. Информация о разработчиках и опыте эксплуатации.

ИС служат для сбора и накопления информации, ее эффективного использования. *Предметная область* – это часть реального мира, подлежащая изучению с целью организации управления и в конечном счете автоматизации. Предметная область представляется множеством фрагментов. Например, для предприятия – отдел кадров, бухгалтерия.

Каждый фрагмент представляется множеством объектов и процессов, использующих объекты, а так же множеством пользователей, характеризующихся единым взглядом на предметную область. Для бухгалтерии объекты – это документы; процессы обработка документов, расчеты.

Каждая СУБД представляет свой обобщенный инструментарий, называемый моделью данных (МД). Поддерживаемые СУБД МД традиционно разбивают на сетевые, иерархические и реляционные. Иногда выделяют простейшую модель данных, называемую плоский файл.

1.4. Модели данных (МД)

Организация данных в СУБД может быть представлена различными моделями данных: сетевой, иерархической, реляционной. Рассмотрим каждую из них.

Организация данных СУБД *сетевого типа* определяется в терминах элемент, агрегат, запись или группа, групповые отношения, БД.

Элемент данных – наименьшая единица структуры данных (рис.1.1).

Агрегат – именованная совокупность элементов или других агрегатов.

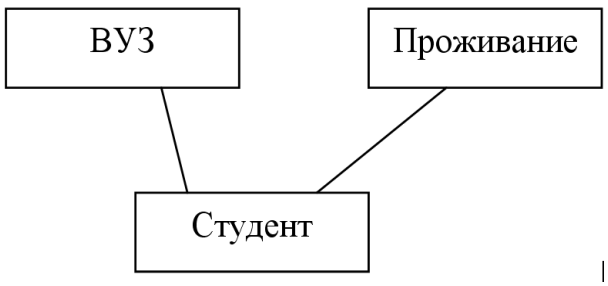


Рис. 1.1. Элементы данных в СУБД сетевого типа

Запись – это агрегат, который не входит в состав никакого другого и составляет основную единицу обработки БД, то есть записи обновляются, извлекаются и удаляются.

Тип записи – определяет состав ее элементов и агрегатов.

Групповые отношения в графическом отображении обозначаются дугами ориентированного графа, в то время как типы записи – вершинами. Такое изображение структуры БД называется диаграммой Бахмана.

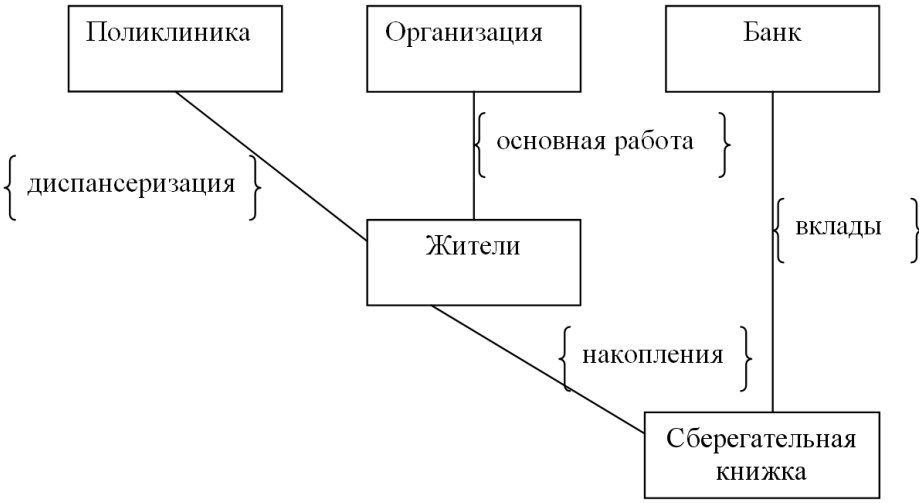


Рис. 1.2. Организация данных в СУБД сетевого типа

Сетевая МД поддерживает БД сетевой структуры (рис.1.2).

Набор операций определяет характер взаимодействия с данными при этом набор является минимальным и достаточным для осуществления операций с данными. Основные операции, допустимые над объектами в сетевой МД:

1. ЗАПОМНИТЬ - позволяет занести запись в БД и автоматически включить ее в групповое отношение, где она объявлена подчиненной с соответствующим режимом включения;
2. ВКЛЮЧИТЬ ГРУППОВОЕ ОТНОШЕНИЕ - позволяет связать запись с определенной записью владельца в установленном групповом отношении;
3. ПЕРЕКЛЮЧИТЬ - меняет запись владельца в том же групповом отношении, например, переключает жителя на обслуживание другой поликлиники;
4. ОБНОВИТЬ - изменяет значение элементов, существующих в БД записи, например, изменение адреса. Перед выполнением соответствующая запись должна быть извлечена;
5. ИЗВЛЕЧЬ;
6. УДАЛИТЬ;
7. ИСКЛЮЧИТЬ ИЗ ГРУППОВОГО ОТНОШЕНИЯ - позволяет разорвать связь между записью владельца и подчиненной для группового отношения, например, если житель прекратил трудовую деятельность, он исключается из группового отношения «основная работа».

Особенности обработки данных в сетевых моделях:

1. Основная единица обработки – «запись».
2. Обработка может быть начата с записи любого уровня, независимо от расположения структуры.
3. От извлеченной записи возможен переход, как к ее подчиненным записям, так и к записям, которым она подчинена.

Сетевые СУБД, поддерживающие сетевую МД – «НИКА», «dBVista».

Структура данных в *иерархической МД* определяется в терминах – элемент, агрегат, запись, групповое отношение, БД.

Отличие иерархической МД в том, что БД может иметь только древовидную структуру. В каждой записи БД существует только один путь от корневой записи, он называется *иерархическим* (рис.1.3). Сетевую структуру можно преобразовать в иерархическую, при этом увеличится число агрегатов.

Иерархическое отношение между записями двух типа являются владельцами

отношений, записи другого подчиненными.

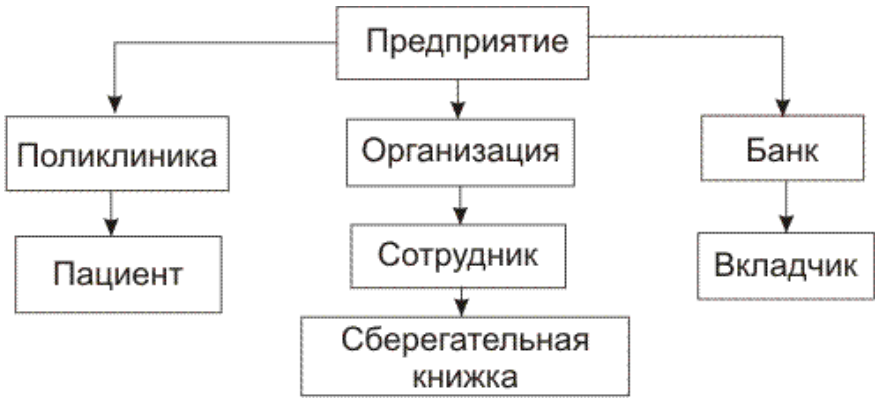


Рис. 1.3. Древовидная структура иерархической МД

Иерархическая МД позволяет осуществлять следующие операции над данными:

1. ЗАПОМНИТЬ;
2. УДАЛИТЬ;
3. ОБНОВИТЬ;
4. ИЗВЛЕЧЬ.

Рассмотрим *реляционную* МД, основанную на понятии «декартово произведение». Пусть D_1, D_2,…,D_n – произвольные конечные множества. Пусть $D_1=\{a_1,a_2\}$, $D_2 = \{b_1, b_2, b_3\}$, тогда декартово произведение равно

$$D_1 \times D_2 = \{a_1b_1, a_1b_2, a_1b_3, a_2b_1, a_2b_2, a_2b_3\}$$

Отношением, определенным на множествах D_1, D_2,…,D_n, называется *подмножество декартова произведения* $D_1 \times D_2 \times \ldots \times D_n$. При этом, множество D_1, D_2,…,D_n называется *доменами отношения*, а элементы декартово произведения $(a_1b_1, a_1b_2, \ldots)$ – *кортежами отношения*. Число *«n»* определяет степень отношения, количество кортежей - его мощность.

Отношения удобно представлять в виде таблицы, при этом строки таблицы соответствуют кортежам, а столбцы атрибутам (рис.1.4). Каждый атрибут определён на некотором домене, фактически представляющим собой множество атомарных значений. Рассмотрим отношения под названием «Студент».

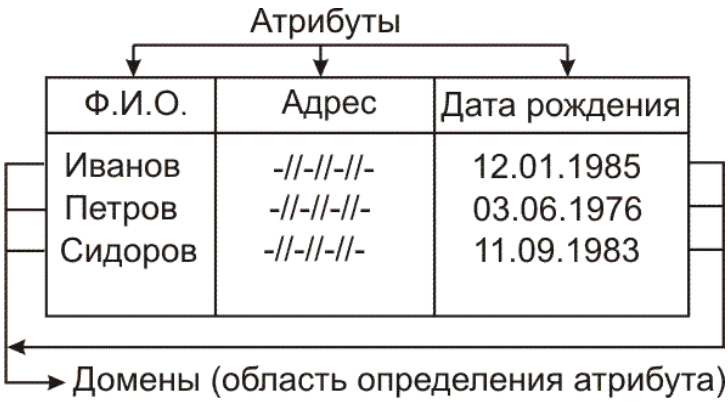


Рис. 1.4. Домены и картежи отношения, представленные в виде таблицы

Модель называется реляционной, потому, что базовым понятием является отношение. Атрибут, значение которого идентифицируют картежи, называется *ключом*. На роль ключа претендует атрибут ФИО (рис.1.4).

В некоторых отношениях картежи идентифицируются конкатенацией значений нескольких атрибутов. В этом случае имеет место *составной ключ*. Например, ФИО + Дата рождения.

Реляционная МД, предусматривает следующие операции обновления БД:

1. ВКЛЮЧИТЬ - требует задания имени отношения и предварительного формирования значения атрибутов нового кортежа. Включение не будет выполнено, если ключ имеет не уникальное значение.
2. УДАЛИТЬ – требует наименование отношения, а так же идентификации ли их группы, подлежащих удалению.
3. ОБНОВИТЬ – выполняется для заданного отношения, при этом можно корректировать как один, так и несколько кортежей отношения.

Основные операции обработки отношений – представляют собой специализированные операции над множествами:

1. ОБЪЕДИНЕНИЕ (C1=AUB): предполагается, что на выходе заданы два односхемных отношения (имеющих одинаковую структуру и атрибуты, но разные названия). Например, объединение двух разных отношений.
2. ПЕРЕСЕЧЕНИЕ (C2=A∩B): на входе два односхемных отношения, выход отношение, постоянное по той же схеме, содержащее картежи из А, которые есть в В. Например, в А студента одной группы сдавали курсовой проект, В другой второй группы, результат – студенты сдавшие курсовой проект.
3. ВЫЧИТАНИЕ (C3=A-B): в результате остаются те картежи из А, которых нет в В. А – информация о всех студентах. В – информация о получивших двойку на экзамене. А-В – информация о студентах успешно сдавших экзамены.
4. ДЕКАРТОВО ПРОИЗВЕДЕНИЕ – (C4=A*B). В отличие от предыдущих операций оно состоит в том, что отношения А (табл. 1.1) и В (табл. 1.2) построены по разным схемам. Схема «А4» включает все атрибуты А и В. В результате отношения попадают сочетание картежей В с А (табл. 1.3).

Таблица 1.1

А-студенты

ФИО	Адрес	Дата рождения

Таблица 1.2

В-отношения экзамена

Дисциплины	Дата	Оценка
УИС	20.08	
ИТУ	26.08	

Таблица 1.3

Экзаменационная ведомость

ФИО	Адрес	Дата рождения	Дисциплина	Дата	0ценка
Иванов	-//-//-//-	1988	УИС	14.06.09	отлично
Сидоров	-//-//-//-	1987	ИТУ	16.06.09	хорошо

Значение атрибута «оценка» могут оставаться пустыми и быть заполнены позднее.

5. ВЫБОРКА (горизонтальное подмножество) – на входе одно отношение. Результат выборки – отношения, построенные по той же схеме, содержащие подмножество исходного отношения, удовлетворяющего условию выборки. Например, студенты, сдавшие сессию на отлично.

6. ПРОЕКЦИЯ (вертикальное подмножество) – на входе операции используют одно отношение.

В результирующем отношении картежи-дубликаты устраняются. Например, проекция отношения студенты на атрибут «дата рождения» получится все даты рождения без повторения.

7. СОЕДИНЕНИЕ - на входе два отношения – А (табл. 1.4) и В (табл. 1.5), построенные по разным схемам. В каждом из них выделяются атрибуты А1 и В1, по которым осуществляется соединение. В отличии от декартова произведения соединение в А произойдет с теми картежами из В, для которых будет выполнено условие А1=В1.

Таблица 1.4

Банки

Номер банка	Адрес банка
1015	
2023	
3456	

Таблица 1.5

Сберегательная книжка

Номер банка	Номер счета	ФИО	Сумма
1015	5010	Иванов	20 000
2023	3456	Петров	50 000

Таблица 1.6

Соединение по атрибуту «Номер банка»

Номер банка	Адрес банка	№счета	ФИО	Сумма
5015		5010	Иванов	20 000

8. ДЕЛЕНИЕ. На входе два отношения А и В. А – экзаменационная ведомость, В-результат. С – результирующее отношение.

В «С» попадут сведения о студентах, которые сдали УИС – отлично, ИСУ – хорошо.

1.5. Нормализация отношений

1.5.1. Понятие нормализации отношений при проектировании реляционных БД

Отношения реляционной БД содержат структурную и семантическую информацию. Структурная информация задается схемой отношений, а семантическая отражает функциональные зависимости между атрибутами (табл. 1.7).

Таблица 1.7

Структурная информация

Текст	Число	Дата	Текст	Дата
ФИО	67890	12.12.78		21.02.88

Таким образом, определенная задача может быть решена с помощью разных структур и семантики информационной части, положенных в основу этих структур, при этом структурная информация задается схемой отношений, а семантическая – выражается функциональными связями между атрибутами.

Фактически структура и семантика отношений является информационной моделью рассматриваемого объекта.

Состав атрибутов отношения БД должен удовлетворять двум основным требованиям:

1. между атрибутами не должно быть нежелательных функциональных зависимостей;
2. группировка атрибутов должна обеспечивать минимальное дублирование данных, их обработку без трудностей;
3. удовлетворение данных требований достигается с помощью нормализации отношений реляционной БД.

Нормализация отношений – это пошаговый обратимый процесс разложения исходных отношений БД на более мелкие и простые отношения. Каждая нормальная форма (НФ) ограничивает типы допустимых функциональных зависимостей отношений.

1.5.2. Первая нормальная форма (1НФ)

Отношения, у которого все атрибуты простые называются приведенной к *первой НФ (1НФ)* (табл. 1.8, 1.9).

Таблица 1.8

Данные о клиенте

Фамилия	Паспортные данные	
	Номер	Дата выдачи

Таблица 1.9

Отношение, приведенное к 1НФ

Фамилия	Номер	Дата выдачи

Существуют различные типы функциональных зависимостей.

Таблица1.10

Данные о преподавателях

Таб. №	Название предмета	Кол-во часов	Фамилия	Должность	Оклад	Кафедра	Телефон
A	B						

Табельный номер и название предмета – составной ключ (табл. 1.10). Если каждое значение атрибута «А» соответствует не более чем одно значение атрибута «В», то говорят, что «В» функционально зависит от «А» ($A \to B$). Например, должность $\to$ оклад, кафедра $\to$ телефон.

Если отношения находятся в 1НФ, то все неключевые атрибуты функционально зависят от ключа, различаться может степень зависимости. Если неключевой атрибут зависит только от части ключа, говорят о частичной зависимости. Например, название предмета $\to$ количество часов.

Может иметь место полная зависимость. Если для атрибутов A, B, C выполняется условие $A \to B$, $B \to C$, а обратная зависимость отсутствует, то говорят, что $A \to C$, C зависит от A *транзитивно*. Например, фамилия $\to$ кафедра $\to$ телефон.

Транзитивная зависимость является более сложным видом зависимости, который нужно учитывать при проектировании структуры БД.

1.5.3. Вторая нормальная форма (2НФ)

Отношения находятся в 2НФ, если находятся в 1НФ и каждый ключевой атрибут функционально полно зависит от составного ключа.

Чтобы устранить частичную зависимость и привести рассматриваемое отношение к 2НФ необходимо разложить его на два отношения так:

1. Построить проекцию без атрибутов, находящихся в частичной функциональной зависимости (табл. 1.11).

Таблица 1.11

Преподаватель

Таб.номер	Фамилия	Дата	Оклад	Кафедра	Телефон

2. Построить проекцию на часть составного ключа, и атрибуты, зависящие от этой части (табл. 1.12).

Таблица 1.12

Сведения о нагрузке преподавателей

Таб. номер	Название предмета	Количество часов

Приведением к 2НФ мы добились устранения дублирования информации вида Преподаватель Предмет 1 Предмет 2.

1.5.4. Третья нормальная форма (3НФ)

В отношении преподаватель имеются транзитивные зависимости:

Таб.номер $\rightarrow$ Должность $\rightarrow$ Оклад. Таб. номер $\rightarrow$ Кафедра $\rightarrow$ Телефон.

Наличие транзитивных зависимостей порождает неудобства:

1. дублирование информации о телефоне для преподавателей одной кафедры;
2. изменение номера телефона кафедры влечет необходимость поиска изменения номеров всех преподавателей;
3. нельзя включить данные по новой кафедре, если в нем нет сотрудников;
4. при увольнении всех сотрудников будут удалены данные по кафедре.

В 3НФ отсутствуют транзитивные зависимости неключевых атрибутов от ключа. Для этого в исходном отношении удаляются последние атрибуты транзитивной зависимости и формируются новые отношения (табл.1.13, 1.14, 1.15).

Таблица 1.13

Преподаватель

Таб. номер	Фамилия	Должность	Кафедра

Таблица 1.14

Должность

Должность	Оклад

Таблица 1.15

Кафедры

Кафедра	Телефон

1.5.5. Усиленная 3НФ и нормальная форма Бойса – Кодда (НФБК)

Рассмотрим отношение курсовой проект (табл. 1.16). С одной стороны, преподаватель по определенному предмету руководит курсовым проектом у студента, с другой стороны – студент выполняет курсовой проект по предмету.

Преподаватель, предмет $\rightarrow$ Студент; Студент $\rightarrow$ Предмет.

Таблица 1.16

Курсовой проект

Преподаватель	Предмет	Студент

Преподаватель + Студент =Составной ключ

Данные отношения находятся в 3НФ, так как отсутствуют частичные и транзитивные зависимости.

Существующие зависимости приводят к следующим отношениям: данные о студенте и его проекте не могут быть занесены в БД, пока не назначен руководитель.

При увольнении преподавателя должны быть удалены сведения о студенте. Устранение данных отношений достигается за счет ликвидации зависимости части составного ключа от неключевого атрибута, то есть зависимости Студент $\rightarrow$ Предмет. Для этого необходимо построить новые отношения без атрибутов, находящиеся в частичной зависимости (табл. 1.17, 1.18).

Таблица 1.17
Руководство

Преподаватель	Предмет

Таблица 1.18
Выполнение

Студент	Предмет

Соединение двух отношений по атрибуту «Предмет» даст исходное отношение «Курсовой проект». Таким образом, отношение находится в 3НФ и отсутствуют частичные зависимости составного ключа от неключевых атрибутов.

Таким образом, процесс нормализации последовательно устраняет:

1. Частичные зависимости неключевых атрибутов от ключа;
2. Транзитивные зависимости неключевых атрибутов от ключа;
3. Зависимости ключей от неключевых атрибутов;
4. Независимые многозначные зависимости (4НФ, 5НФ).

1.5.6. Место процесса нормализации

Вначале изучают предметную область и составляют исходные отношения, затем производят нормализацию. Полученные отношения описывают средствами СУБД и вводят в ЭВМ.

Следует отметить, что нормализация является необходимым элементом проектирования БД, поэтому на практике стараются формировать исходные отношения в БД сразу в нормализованном виде.

Нормализация отношений, с одной стороны, приводит к увеличению числа элементов в структуре данных, поэтому возрастает ее сложность, с другой стороны, благодаря устранению дублирования данных ускоряются операции доступа к ним, и уменьшается объем памяти, необходимых для их хранения.

1.5.7. Виды отношений между таблицами при проектировании структуры БД

Структура реляционной БД подразумевает наличие таблиц и связей между ними:

1. *«Один-к-одному».* Каждой записи одной таблицы соответствует одна и только одна запись другой и наоборот (рис.1.5). Связь между таблицами осуществляется на основании значений, выбранных для связи полей.

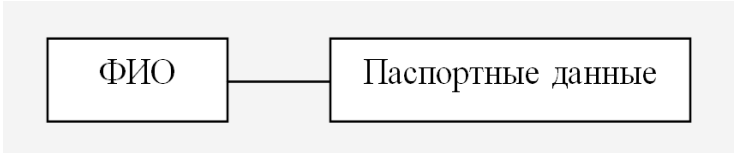


Рис. 1.5. Связь «один-к-одному»

Связь «один-к-одному» не является ошибочной, но применяется редко и чаще всего представляет собой упущенные атрибуты одно и той же таблицы, то есть на практике вместо связей «один-к-одному» формируют одну таблицу с соответствующими атрибутами.

2. *«Один-ко-многим».* Каждой записи в первой таблице соответствует одна или более записи во второй, если связь не жесткая, то может соответствовать (рис.1.6).

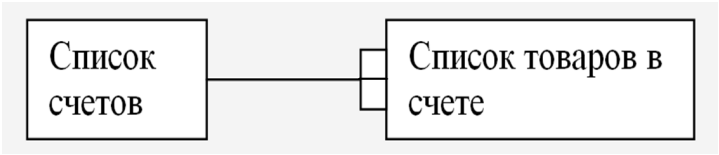


Рис. 1.6. Связь «один-ко-многогим»

Каждому счету в списке соответствует один и более товар (рис.1.7).

Рис. 1.7. Соединение таблиц связью «один-ко-многоим» по номеру счета

Связь «один-ко-многим» является основной связью при формировании отношения вида «список – расшифровка списка».

3. *«Многие-к-одному».* Фактически связь «один-ко-многим», но рассматриваемая со стороны многие. Связь интерпретируется: каждой записи первой таблицы должна соответствовать одна и только одна запись во второй.

Рис. 1.8. Структура данных о сотрудниках

Очень часто связь «многие-ко-одному» используется в форме ссылки со стороны многие по ходу на соответствующий код со стороны один, при этом таблица со стороны один является справочником, возможно заполнением заранее. Если таблица со стороны один заполнена, то связь со стороны «многие» жесткая, обязательная, а со стороны «один» – необязательная (рис.1.8.).

4. *«Многие-ко-многим».* Каждая запись первой таблицы может (должна) быть связана с одной и более записей во второй и наоборот (рис.1.9).

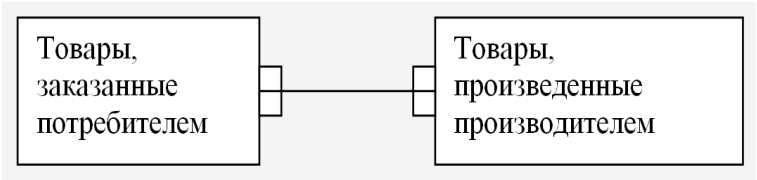


Рис. 1.9. Связь «многие-ко-многим»

Если связь «многие-ко-многим» не жесткая, то она подлежит уточнению и преобразованию в две связи «один-ко-многим», если связь жесткая, то она всегда ошибочна.

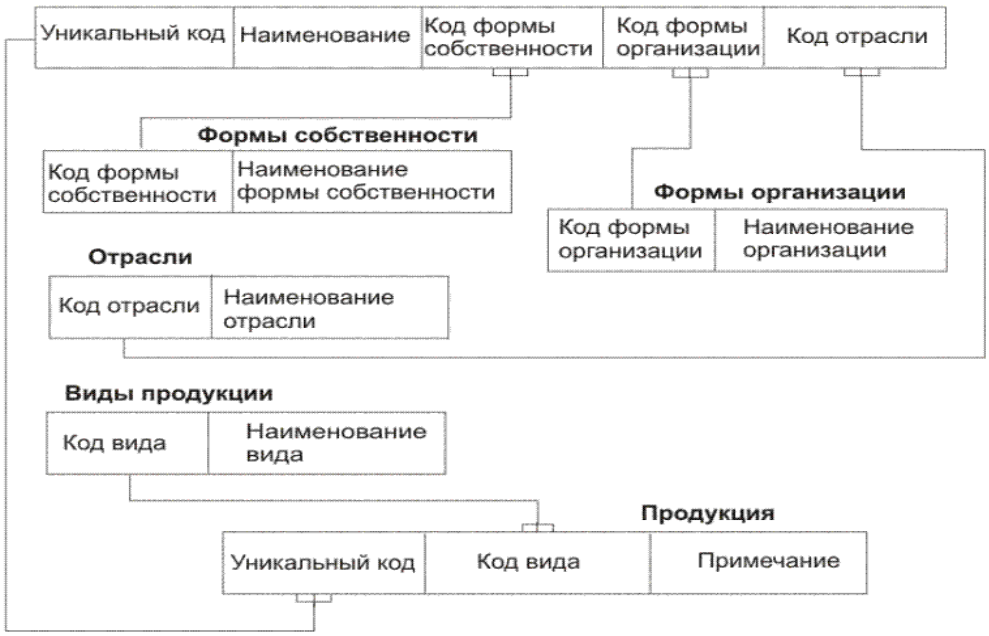


Рис. 1.10. Структура БД «Информация о предприятиях»

Рассмотрим структуру БД «Информация о предприятиях», представленную на рис. 1.10. Формы собственности: государственная, частная и т.д. Формы организации: ООО, ОАО, ЗАО и т.д. Отрасли: легкая промышленность, машиностроение и т.д. Каждое предприятие может производить свою номенклатуру продукции.

В таблицах первое поле – это уникальный идентификатор, код. По нему создан уникальный индекс (повторение не допускается). Поле со стороны «многие» также может быть проиндексировано неуникальными индексами (повторение допускается). В таблице «Производимая продукция» уникальным идентификатором (ключом) может быть уникальный код - код вида.

Как правило, поля, предназначенные для связи делают числового типа. Фактически связь происходит с помощью индексных выражений для значений выбранных для связи полей, всякий раз, когда указатель записи выбранной таблицы перемещаются, перемещаются на соответствующие значения указатели в связанных таблицах.

Рассмотрим структуру данных информационной системы о студентах вуза, изображенную на рис. 1.11.

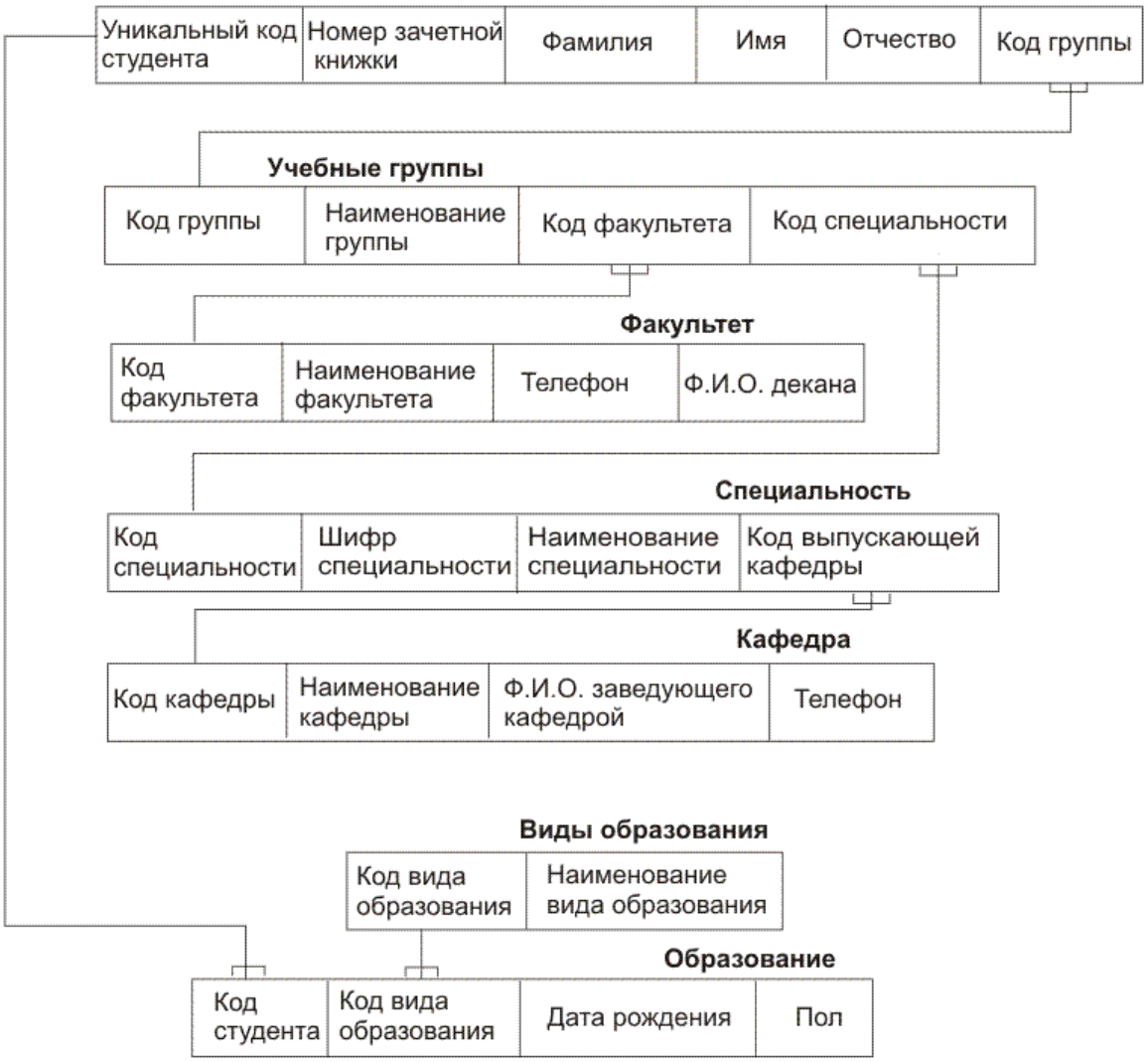


Рис.1.11. Информационная система о студентах вуза

В представленном варианте предполагается, что специальность может находиться на разных факультетах. Если исходить из того, что специальность находится только

на одном факультете, то поля «Код факультета» необходимо перенести в таблицу «Специальность».

1.6. Общая характеристика СУБД Access

1.6.1. Типы данных в СУБД Access

БД Access – совокупность взаимосвязанных таблиц, состоящих из полей. Поля могут быть следующих типов:

1 Текстовый – до 255 символов;

2 Namber (числовой) – для установки подтипа числовых данных служит свойство – размер поля (Field Size).

3 Date/Time – дата/ время.

4 Curency – денежный;

5 AutoNamber (счетчик) – уникальный номер, определенный для каждой записи либо случайно, либо путем увеличения на один.

6 Logic (логический);

7 Memo – до 65535символов – блоки данных хранятся отдельно;

8 Поле ссылки на объект OLE;

9 Hypelink – гиперссылка;

Для полей, имеющих один из этих типов, можно задавать свойства:

1.Caption –подпись;

2.Required (обязательное поле);

3.Формат поля – способ отображения данных из поля;

4.Input Mask (маска ввода);

5.Индексированное поле- Indexed – создается ли по полю индекс?

Существует два режима:

1. Daplicates Ok – совпадения допускаются;
2. No daplicates – не допускаются

Значение по умолчанию – автоматически добавляется, если пользователь не ввел другое;

Проверка данных, вводимых в поле (Validation Rule)При несоответствии вводимых данных заданному условию выдается определенное пользователем сообщение об ошибке.

1.6.2. Методы упорядочивания данных в СУБД Access

1. Физическая (сортировка) – прядок следования записей при сортировке становится таким, как требует условие сортировки. Данный порядок становится реальным (физическим) в большинстве СУБД при сортировке создается новая таблица. Использование сортировки приводит к неудобствам:

- при создании новой таблицы можно столкнуться с нехваткой места;
- при наличии нескольких таблиц бывают несоответствия данных в них;
- при вводе новых данных нужна новая сортировка.

2. Логическое (индексирование). В Access для обеспечения логического упорядочения применяют следующие виды индексов:

- простой индекс – это тот, который создается по одному столбцу;
- составной – создается по нескольким столбцам.

Упорядочение таблицы происходит в соответствии с активным индексом.

В одной таблице можно создать несколько индексов по различным признакам и, делая активным определенный индекс, можно менять признаки, по которым упорядочиваются таблицы.

Количество создаваемых индексов точно соответствует количеству признаков, по которым надо упорядочить.

Механизм упорядочения таблицы с помощью индекса заключается в создании для каждого индекса связанного списка, который «руководит» порядком следования записей в таблице.

Следует учитывать, что при добавлении, удалении записей или обновлении значений в индексном столбце требуется обновлять индексы, таким образом, при большом количестве индексов и повышенном объеме таблицы, индексные выражения могут занимать много места на диске и для их обновления требуется определенное время.

Индексировать можно любые поля, кроме MEMO, ссылок на OLE объект и гиперссылок.

Индексы бывают:

1. Уникальные (No daplicates);
2. неуникальные (Daplicates).

В уникальном индексе не допускаются повторения индексного выражения значений полей.

Ключевое поле предназначено для уникальной идентификации записи в таблице, оно автоматически индексируется и, свойству индексированное поле присваивается значение No Daplicates.

1.6.3. Связывание таблиц в СУБД Access и обеспечение целостности данных

Для связывания таблиц используется схема данных Relationships.

Удаление таблицы из схемы данных не означает удаление ее из БД. При связи таблицы необходимо соблюдать правила соответствия типов полей (индексов), используемых для связи. Для связи можно использовать типы «один-ко-многим», «один-к-одному», связи «многие-ко-многим» в Access не допускаются. При установлении связи главной считается таблица один, другие - дочерняя и подчиненная.

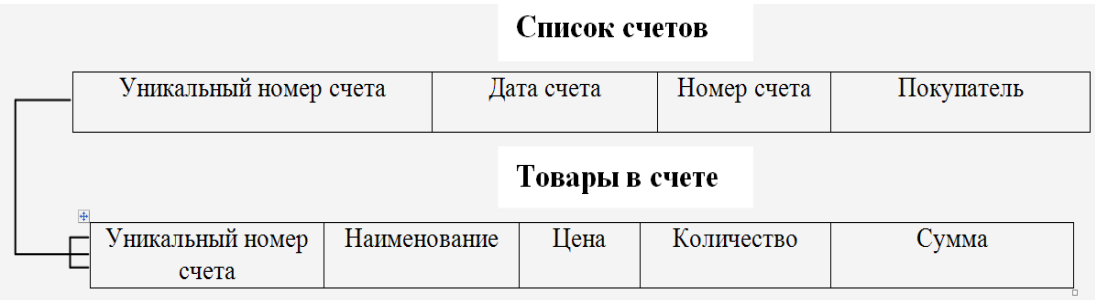


Рис.1.12. Соединение таблиц связью «один-ко-многоим» по номеру счета

Рассмотрим отношение, представленное на рис. 1.12. Код счета в списке счетов проиндексирован уникально (Daplicates). Уникальный код счета в таблице «Товары» проиндексирован не уникально (No Daplicates).

Целостность данных – логическая непротиворечивость данных одной и другой таблицы, то есть, если целостность данных «обеспечена», то в таблице «Счета» нет записей, для которых отсутствуют подчиненные записи в таблице «Товары» и наоборот.

Если удалить запись в списке счетов, то для товаров не будет ссылки на соответствующий счет, то есть будет нарушена целостность данных.

Задание обеспечения целостности данных в Access позволит контролировать, что:

1. Невозможно ввести в связанное поле подчиненной таблицы значение, отсутствующее в соответствующем поле главной, но можно ввести пустое значение, показав, что для данной записи связь отсутствует.

2. Невозможно удаление записей из главной таблицы, если существуют связанные с ней записи в подчиненной.

3. Невозможно изменить ключевое поле главной таблицы, если в подчиненной существуют связанные с ней записи.

Чтобы преодолеть ограничения на изменение связанных записей, сохранив при этом целостность, необходимо установить флажки:

1. Каскадное обновление связанных полей;
2. Каскадное удаление связанных полей.

Если установить каскадное обновление, то при изменении ключевого поля главной таблицы автоматически будут изменены значения полей связанных записей.

Если установлено каскадное удаление, то при удалении записей в главной таблице удаляются все связанные записи в подчиненной.

Контроль ввода данных необходим, если нужна твердая гарантия того, что пользователь не ввел недопустимых значений данных.

Можно задать условие на значение поля или на значение записи.

1.6.4. Сортировка, поиск и фильтрация данных в Access

Когда таблица открывается в режиме таблица – она упорядочена по значению ключевого поля, если ключевое поле не определено, то в порядке ввода записи.

Для сортировки необходимо установить курсор на столбец и выбрать тип сортировки:

1. По возрастанию (Sort Ascending);
2. По убыванию (Sort Descending).

Поиск. Наиболее быстро происходит для поля имеющего индекс, так как фактически сначала ищется соответствующий ключ в индексе, а затем осуществляется переход к соответствующей записи.

Самый медленный вид поиска – это поиск на совпадение с любым произвольным полем (без индекса). Данный поиск происходит медленно, так как осуществляется последовательно, начиная с первой записи.

Фильтрация данных: позволяет показать или скрыть записи удовлетворяющие, неудовлетворяющие определенному условию.

Признаком того, что в таблице отображаются отфильтрованные записи является признак ФЛТР в строке «состояние приложения».

В Access имеются следующие способы фильтрации:

1. Фильтр по выделенному фрагменту;
2. Обычный фильтр по значению нескольких полей;
3. По значению фильтр “Для» - Filter For – здесь задается значение для фильтра;
4. Расширенный фильтр.

Отличительные особенности фильтров:

1. Фильтры не позволяют объединять таблицы;
2. Фильтры не дают возможности включать в результирующую таблицу выбранные поля. Отображаются все поля таблицы;
3. Результат фильтрации не может быть сохранен как объект в окне БД;
4. Фильтры не позволяют осуществлять ряд действий – нахождение суммы, поиск среднего значения, подсчет количества записи.

Формы. Формы – средства ввода и отображения данных. Они содержат средства управления, помощью которых осуществляется доступ к данным. Элементами управления являются текстовые поля для ввода и редактирования данных, флажки,

переключатели, списки, подписи. Режим «формы» предоставляет больше возможностей для редактирования, чем режим «таблица».

Отчеты. Они позволяют выбрать из БД требуемую информацию и оформить ее в виде документов, которые можно просмотреть или напечатать. В отличие от форм, отчеты не предназначены для ввода данных, но в них можно применять элементы управления.

1.7. Архитектура информационной системы

1.7.1. Уровни представления, поддерживаемые СУБД

Восприятие ИС конечным и внутренним пользователем различно. В современной ИС поддерживается несколько уровней представления (абстракции), которые принято ассоциировать с понятием архитектуры ИС (рис.1.13).

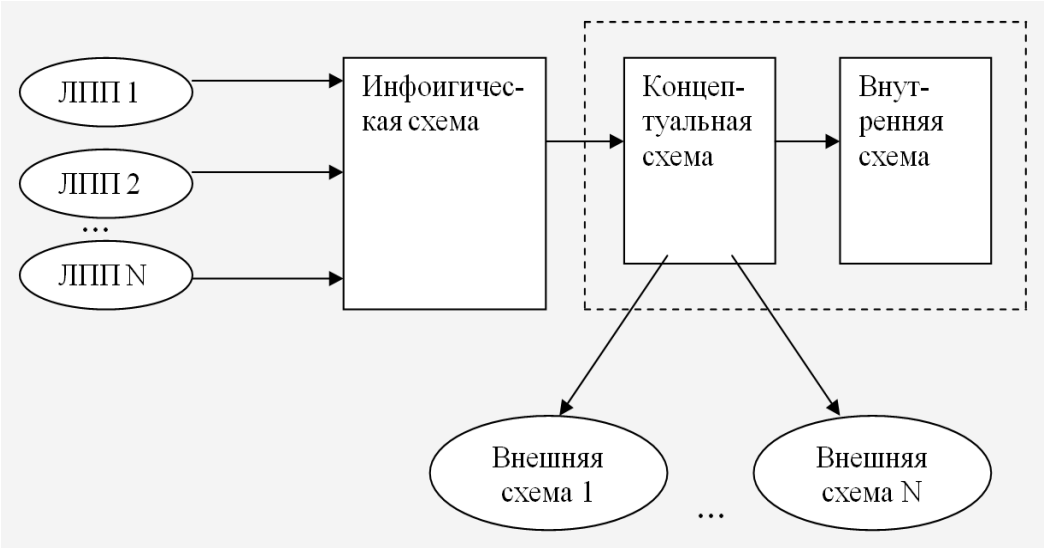


Рис. 1.13. Уровни абстракции, поддерживаемые СУБД

Начальный уровень абстракции соответствует представлениям о предметной области конечных пользователей – *локальные пользовательские представления (ЛПП)*.

Инфологический уровень – второй. Он представляет собой интеграцию ЛПП, соответствующую взгляду на предметную область ее администратора. Администратор видит все множество информационных объектов, все возможные ассоциации между ними, в то время как каждый пользователь видит лишь ограниченный объем предметной области.

ЛПП и инфологическое представление определяет информационные потребности, но не затрагивает вопроса – как указанные данные будут представлены в памяти ЭВМ.

Концептуальный уровень – соответствует представлению о логической организации данных администратора БД. На данном уровне существует привязка к модели данных и средствам ее реализации. Ряд авторов считают, что на концептуальном уровне целостность является свойством структуры данных адекватно отображать особенности предметной области.

На концептуальном уровне строятся ER-модели, разрабатывается описание БД в терминах и ограничениях конкретного информационного средства.

Внешняя схема является отображением соответствующей части ИС, чувствительной для определенного ее фрагмента, то есть пользователь работает только с той частью системы, которая ему нужна.

Представления данных в памяти ЭВМ и организация их хранения определяет внутренний уровень абстракции.

Параметры внутреннего уровня влияют на эффективность ИС в виде объема памяти, времени реакции системы.

1.7.2. Требования к проектированию БД. Языковые (лингвистические) и сервисные средства описания данных

1. Корректность схемы БД, то есть соответствие моделируемой предметной области. В экономической литературе это свойство называется целостностью.
2. Обеспечение ограничений, таких как конфигурация вычислительной системы, ресурсы внешней и оперативной памяти.
3. Эффективность функционирования – обеспечение требуемого времени реакции на запросы и обновление данных.
4. Защита данных от разрушений при сбоях оборудования, некорректных обновлений и если необходимо, несанкционированного доступа.
5. Простота и удобство эксплуатации.
6. Гибкость – возможность последующего развития и адаптация системы к изменениям предметной области и новым требованиям пользователей. Для принятия проекта необходимо соблюдать первые четыре требования.

Важная особенность БД – это отделение данных от программ и обработки, поэтому в ИС описание данных задаются не языком программирования, а с помощью оригинальных средств спецификации данных.

На каждом уровне абстракции используются свои языки.

Для обработки информации используются языки манипулирования данными:

1. Специализированные языки манипулирования данными предназначены для описания и обработки данных в ИС, и по сути являются языками универсальных СУБД. Например, XBax, Oracle? SQL.
2. Универсальные языки – Си, Паскаль – используются для обработки данных, при этом непосредственно для манипулирования данными существуют библиотеки.

При этом обработка данных осуществляется с помощью обращения к процедурам и функциям на универсальном языке программирования.

1.7.3. Построение схемы БД

На концептуальном уровне описывается организация данных и ограничение целостности. Логическая организация определяет перечень типов записей, а так же установление отношений между ними.

Ограничение целостности представляют собой процедуру проверки достоверного состояния БД, так же проверку областей допустимых значений полей.

Описание внутреннего уровня – это стратегия хранения данных в памяти, то есть разбиение на файлы, организация файлов. Внутренний уровень еще называют *физическим*.

В большинстве СУБД описание логических и физических уровней совмещены, и такое описание в целом называется *схемой данных*. Для описания схемы существует специальный язык.

Подсхема – это описание данных на внешнем уровне представления. Фактически эта часть схемы чувствительная для одного или нескольких приложений. В такой схеме одна внешняя запись может представляться из нескольких записей концептуального уровня. В записях могут присутствовать виртуальные поля, то есть те, которые фактически не хранятся в БД, а вычисляются алгоритмически.

1.7.4.Страничная организация доступа данных

Страница – участок памяти для хранения подмножества данных.

В большинстве систем в начале страницы размещается служебная запись, называемая заголовком страницы. Она используется для управления содержимым страницы.

В странице имеются несколько записей и свободный участок для хранения новых. Страницы, как правило, нумеруются, и СУБД умеет их извлекать по номерам.

Хранимая запись – состоит из служебной и информационной частей.

Служебная часть используется СУБД для идентификации записи, хранения признака логического удаления и кодирования значения элемента. Никакие пользовательские программы не имеют доступа к служебной части.

Информационная часть – это сами данные. Каждой записи БД система присваивает внутренний идентификатор, называемый КБД (ключ БД). Например, номер записи в файле.

На практике применяются два способа *удаления записи*:

1. *Логическое* – запись реально остается в таблице, а в служебной части помечается как удаленная.

Достоинства:

- высокая скорость;
- возможность оперативного удаления и восстановления.

Недостаток – накопление мусора

2. *Физическое* – реальное удаление записи из таблицы.

При физическом удалении происходит одновременно переформирование файла таблицы и перестройка индексов.

Достоинства – таблица реально очищается от записи, то есть, нет накопления мусора.

Недостаток – невозможность осуществления операции в распределенном режиме работы; медленно.

На практике удаление осуществляется так – сначала записи удаляются логически, а затем физически. Хорошей практикой является помещение удаленных записей в архив.

Доступ к БД обеспечивается следующими способами:

1. Последовательная обработка БД. Достоинство – универсальность; недостаток – медленно.

2. Доступ по ключу – зная ключ, можно быстро извлечь запись.

3. Доступ по структуре – доступ от одной записи участницы группового отношения к другой, в соответствии со структурой БД.

Сократить время поиска записей и обеспечивать их упорядочение позволяет *индексирование*:

1. *Индексные таблицы*. При использовании индексных таблиц записи предварительно должны быть отсортированы по возрастанию значений ключей. Цель – найти как можно быстрее запись с заданным ключом. Записи располагаются на страницах.

Изобразим процесс формирования индексных таблиц. Пусть ключи имеют значение чисел натурального ряда (рис.1.14).

После формирования страницы формируется строка таблицы, называемая младший индекс, она ставит в соответствие номер страницы и максимальное значение ключа на этой странице.

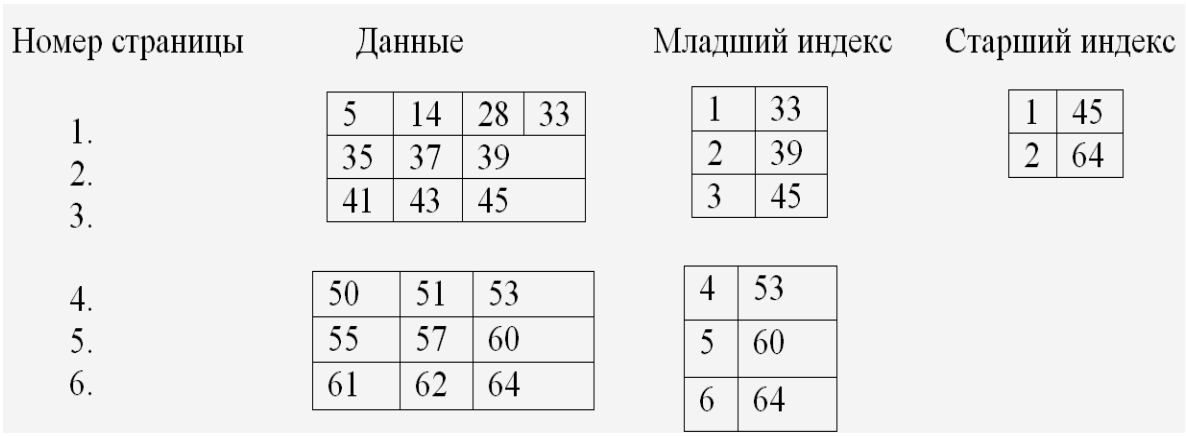


Рис. 1.14. Индексные таблицы

После формирования страницы, обслуживаемых младшим индексом, формируется строка в таблице, называемая старший индекс.

Поиск необходимой записи осуществляется по ключу следующим образом – в старшем индексе ищется строка с ключом большем или равном заданному. Далее находим по номеру соответствующий младший индекс, просматривая найденный младший индекс, находим страницу. На странице поиск последовательный.

2. *Бинарные деревья.* Индексная таблица при этом методе индексирования состоит из ключей и адресных ссылок (рис.1.15). Здесь нет необходимости предварительно упорядочивать записи.

Бинарное дерево формируется следующим образом – первая запись становится корневой. Если ключ новой записи больше чем у текущей величины, то осуществляется переход вправо, иначе – влево, пока не будет найдена конечная величина.

Номер записи	Ключ
1	15
2	17
3	8
4	7
5	19
6	20
7	21

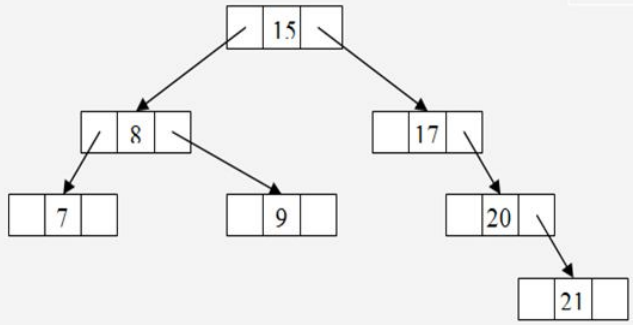


Рис. 1.15. Бинарные деревья

Бинарные деревья обладают существенным недостатком – это проявляется в несбалансированности бинарного дерева, то есть конечные вершины здесь могут оказаться на разных уровнях иерархии.

Иногда может оказаться так, что определенная последовательность данных приведет к построению дерева с одним иерархическим путем, что не даст никаких выигрышей в поиске.

В связи с этим применение бинарных деревьев серьезно ограничено на практике.

Каждое Би-дерево порядка *«n»* состоит из *«n-1»* ключей и *«n»* адресных ссылок. Отличие Би-дерева от бинарного дерева заключается в его сбалансированности.

Бинарное дерево растет вниз и корневая вершина не меняется. Би-дерево растет вверх и корневая вершина меняется.

В настоящее время для организации индексов используются модифицированные би-деревья.

3. *Вторичное индексирование.* Если первичное индексирование ставит в соответствие номеру ключа номер страницы, где находится запись или, с помощью ссылок определенным образом упорядочивает записи, то вторичное индексирование предназначено исключительно для выборки записей и не влияет ни на их расположение, ни на их организацию в целом. Вторичные индексы называют *инвертированными списками* (табл.1.19).

Таблица1.19

Инвертируемый список

Должность	КБД
Инженер	1, 15, 18
Техник	2, 7, 9, 10

Для создания инвертированных списков существуют сервисные программы, создающие их в требуемый момент времени.

1.8. CASE-технологии в разработке информационных систем (Computer Aided Soft ware Engineering)

1.8.1. Понятие и назначение CASE-технологий

CASE – автоматизированный инжиниринг программных средств, то есть совокупность методик проектирования и сопровождения программных средств на всем из жизненном цикле, поддержанная взаимоувязанными средствами автоматизации.

Современные технологии разработки ИС поддерживают следующие стадии жизненного цикла ИС:

1. стратегическое планирование (стратегия);
2. анализ;

3. проектирование;
4. реализация;
5. сопровождение.

Стратегическое планирование – за короткое время изучается деловая деятельность заказчика и формируется общее представление о ней.

Здесь обосновывается необходимость разработки новой системы.

В течение этой стадии разрабатываются общие или укрупненные модели предметной области.

Анализ – в течение которого разрабатываются детальные модели предметной области, формируются точные требования к будущей программной системе и закладывается точная основа для перехода к проектированию.

Проектирование – стадия точного плана реализации требований, определенных ранее, если на предыдущих этапах определилось: Что должно быть сделано? – то на этапе проектирования – Как это должно быть сделано ?

Реализация – стадия, когда выполняется непосредственная реализация и сборка программной системы.

Сопровождение – состоит в том, что ИС поддерживается в актуальном состоянии, если в связи с извлечением в предметной области потребуются значительные доработки, весь жизненный цикл повторяется сначала.

1.8.2. Методы разработки структуры БД: ER-моделирование

ER-моделирование – метод разработки ER-модели, отражающей информационные потребности предприятия-заказчика и, не зависящей от физической реализации хранилищ данных и метода доступа к ним.

Разработка реляционной таблицы, то есть модели данных, ориентированных на применение реляционной СУБД.

Технологические операции в ER-моделировании

1. *Разработать ER-модель* – на выходе произвольное описание деятельности предприятия заказчика в форме, стандартизированном для соответствующего метода ER-моделирования.

2. *Получение ER-диаграммы* – информация, позволяющая работать с графическими образами ER-модели. Получаем более наглядные графические отображения предметной области. В настоящее время этапы 1 и 2 совмещены.

3. *Проверка качества ER-модели* – осуществляется автоматическим CASE-средством, кроме этого, возможна экспертная проверка качества в соответствие с заданным списком вопросов.

4. *Генерация реляционных таблиц из ER-модели.* В соответствие с выбранным СУБД осуществляется генерация таблиц или программного когда для их создания и связывания. Полученные таблицы в ряде случаев могут требовать доработки.

CASE-технологию можно использовать для модификации или документирования существующей БД, такая задача возникает на стадии сопровождения, при необходимости внесение изменений в ИС и называется *реинжинирингом структуры БД*.

После импорта физических файлов производится операция по доработке сущностей и формированию связей.

В настоящее время существуют несколько различных методов построения *ER-модели*, являющихся продолжением классического метода, разработанного П. Ченом.

Рассмотрим метод, разработанный фирмой Oracle, описанный в книге R. Barker. ER-модель описывает совокупность важных объектов предметной области сущностей (Entity), их свойств (атрибутов) и отношений между объектами (связей) – Relationship. ER-модель еще называют моделью «Сущность –связь».

ER-диаграмма является графическим отображением ER-модели и, состоит из следующих базовых элементов:

1. Изображение сущности в виде прямоугольного блока с именем сущности, записанного внутри прописными буквами (рис.1.16).

Рис.1.16. Изображение сущности

2. Связь, изображаемая в виде линий, соединяющих два блока сущностей (рис. 1.17(а)) или один блок рекурсивно сам с собой (рис. 1.17(б)). Для каждого конца связи определяется его имя, обязательность и множественность.

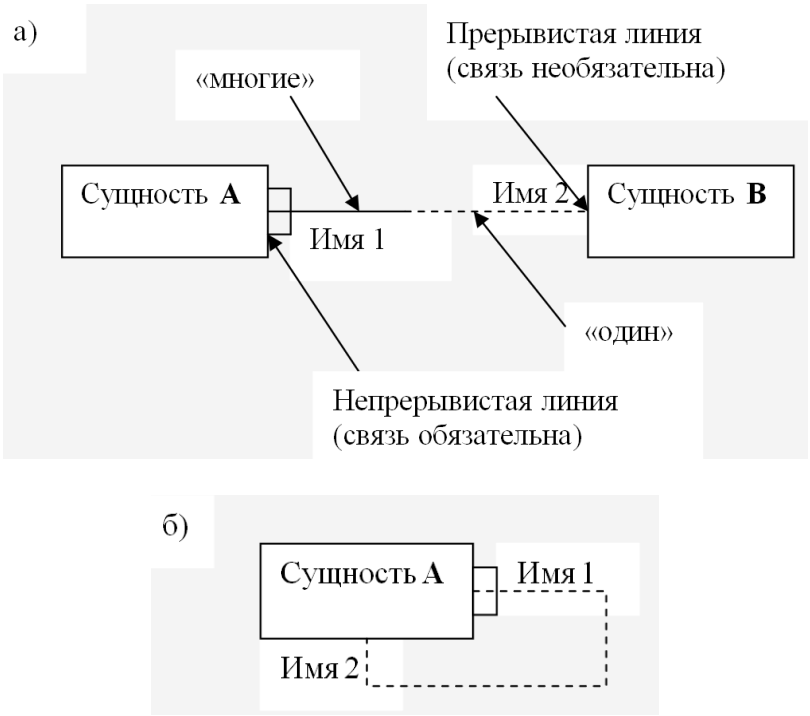


Рис.1.17. Изображение связи: а) связь, соединяющая два блока; б) связь, соединяющая один блок рекурсивно

Читается ER-диаграмма с точки зрения выбранной сущности, отдельно для каждого конца связи, так показывается, как выбранная сущность связывается с другими сущностями ER-модели (рис. 1.18).

Рис. 1.18. Соединение сущностей «Студент» и «Группа»

При чтении ER-диаграммы необходимо учитывать обязательность и множественность выбранного конца связи. Если конец связи является обязательным, то перед именем связи добавляется фраза «должен», «должен быть». Если конец связи является не обязательным, то пред именем связи добавляется фраза «может», «может быть».

Например, каждый студент должен учиться в одной и только одной группе. С другой стороны – каждая группа может состоять из одного и более студентов. Дополнительные соглашения в ER-моделировании могут касаться ограничений по объему сущностей.

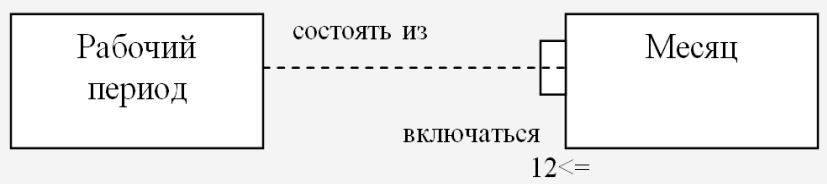


Рис. 1.19. Соединение сущностей «Рабочий период» и «Месяц»

Каждый <имя сущности> должен или может <имя конца связи> один и только один или один и более <имя сущности 2> (рис.1.19).

Если указаны ограничения по объему сущности, то их необходимо учитывать при чтении ER-модели, для каждой сущности задаются атрибуты, которые могут быть обязательные (*) и не обязательные (о) (рис. 1.20).

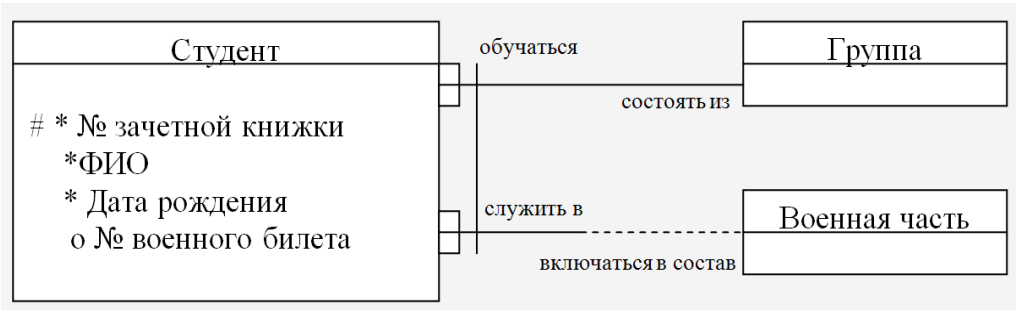


Рис. 1.20. Использование ограничений по объему сущности

Для уникальной идентификации экземпляров сущности необходимо наличие специального атрибута, называемого ключом и, обозначаемого «#». Он указывается первым по списку атрибутов. Ключ обязательно должен состоять из обязательных атрибутов. Каждый атрибут в ER-моделировании может быть определен на домене.

Домен описывает множество значений, возможный их диапазон, формат и тип, то есть фактически домен является областью значений атрибута, например, домен для атрибута «Номер зачетной книжки» может задавать диапазон номеров и формат их значения (рис. 1.21).

Для каждой сущности могут быть заданы альтернативные сущности, при этом каждый экземпляр выделяемой сущности может быть связан с экземпляром одной из альтернативных сущностей.

Читается ER-модель с альтернативными сущностями аналогично обычной, но с помощью или указываются связи с альтернативными сущностями.

Для указания альтернативных связей используются исключающие дуги, она должна пересекать концы связей одинаковой множественности, обязательности, если конец связей имеет обозначение ключей идентификации, то альтернативный конец

связи так же должен быть с ключевой идентификацией. Практика ER-моделирования показала, что альтернативными должны быть не более трех сущностей.

Рис. 1.21. Соединение сущностей «Сотрудник» и «Отдел»

В ER-моделировании существует понятие подтипа. При использовании подтипов существует правило, согласно которому множество подтипов должно полностью описывать всю совокупность экземпляров сущности. Поэтому при описании подтипов необходимо использовать подтип с именем другой имя сущности.

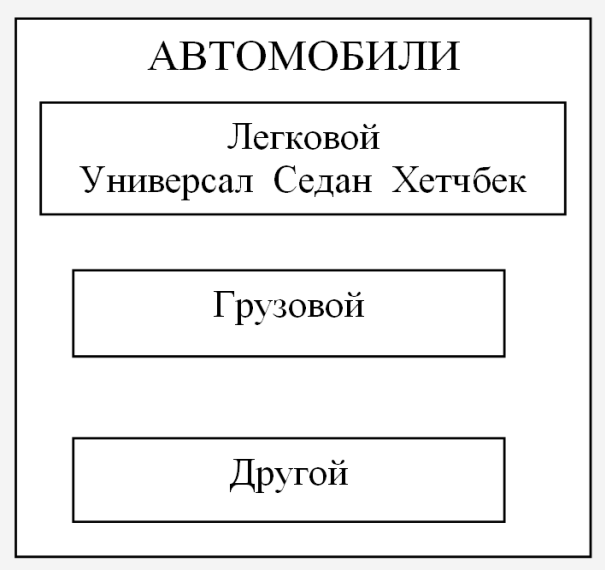


Рис.1.22. Супертип «Автомобили»

Опыт ER-моделирования показал, что уровень вложенности подтипов не должен быть больше двух. Нормальным считается количество подтипов – 3-4.

На рис. 1.22 представлена сущность – супертип: «Автомобиль, который может быть легковым или грузовым или другим».

1.8.3. Семантика связей в ER-моделировании

По виду употребляемой связи можно сделать заключение о ее применяемости. Рассматриваемые связи могут быть допустимыми, часто и редко встречаемыми и невозможными.

1. *Связи типа «многие-к-одному».*

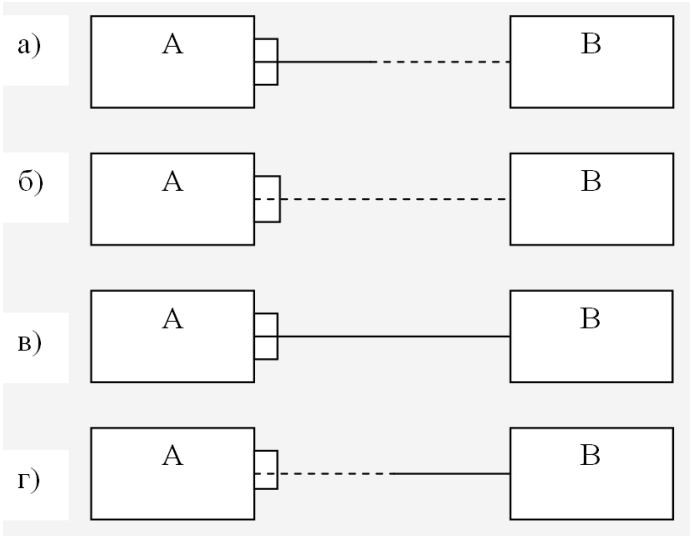


Рис. 1.23. Связи типа «многие-к-одному»

Тип связи изображенный на рис. 1.23(а) встречается часто и является основным видом связи при построении ER-модели.

Связь, представленная на рис.1.23(б), говорит, что конкретные экземпляры А и В могут существовать без связи между ними, она применяется редко и требует уточнения.

Связь, представленная на рис.1.23(в), является сильной и используется при описании взаимоотношений типа «Счета-товары».

Тип связи, изображенный на рис.1.23(г), встречается очень редко, при ближайшем рассмотрении такие связи часто становятся связями «многие-ко-многим».

2. *Связи типа «один-к-одному»*

Связи, изображенные на рис.1.24, встречаются очень редко. В целом связи «один-к-одному» часто представляют собой упущенные атрибуты одной и той же сущности. Поэтому на практике от них стараются избавиться, добавляя атрибуты в базовую сущность.

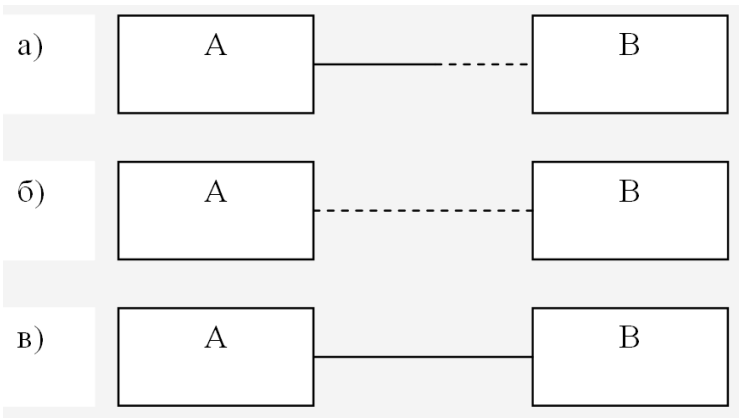


Рис.1.24. Связи типа «один-к-одному»

3. *Связи «многие-ко-многим».*

Тип связи, изображенный на рис. 1.25 (а), встречается редко, в дальнейшем должна пересматриваться и детализироваться.

Связь, представленная на рис. 1.25 (б), может существовать, но для реализации конкретного проекта БД она должна уточняться и представляться в виде двух связей, возможно с помощью дополнительной таблицы.

Другой тип связи «многие-ко-многим» изображен на рис. 1.25.(в). Данная связь невозможна на практике, она всегда неправильна.

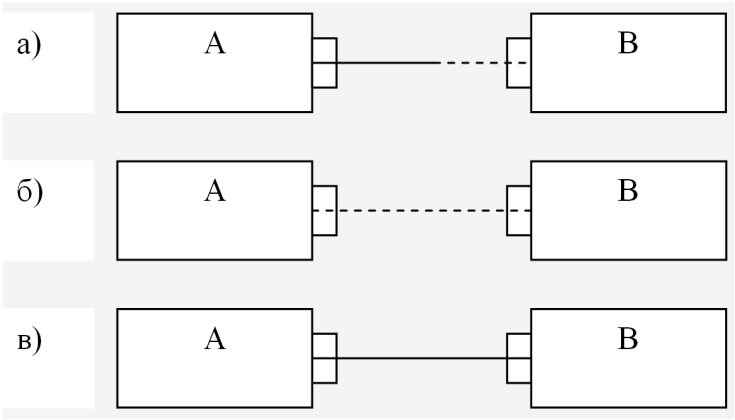


Рис. 1.25. Связи «многие-ко-многим»

Предположим, что студент обучается в двух группах (рис.1.26,1.27).

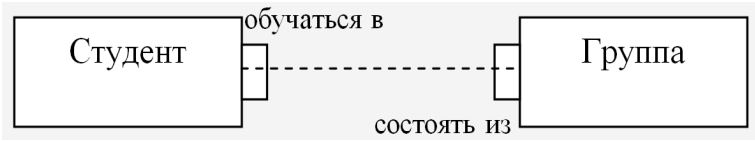


Рис. 1.26. Отношение сущностей «Студент» и «Группа»

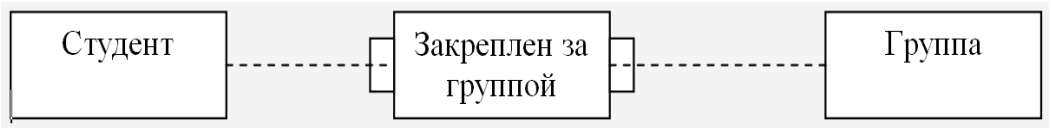


Рис.1.27. Отношение сущностей «Студент» и «Группа»

4. *Рекурсивные связи типа «многие-к-одному»*

Изображенные на рис.1.28 связи невозможны, так как не правильны.

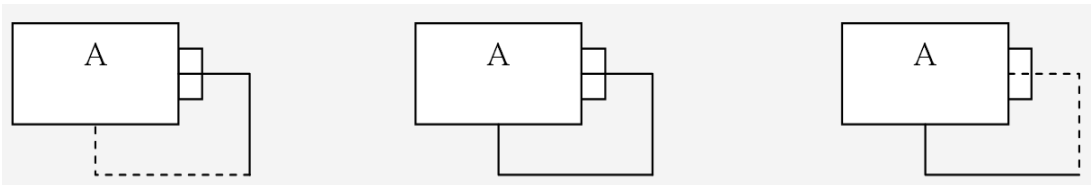


Рис. 1.28. Неправильные рекурсивные связи типа «многие-к-одному»

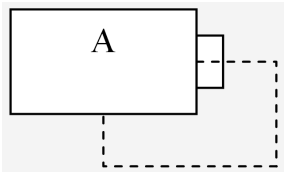


Рис.1.29.Допустимые рекурсивные связи типа «многие-к-одному»

Допустимым является только один тип связи «многие-к-одному» (рис.1.29), используемый для отображения иерархических отношений, таких как организационная иерархия, классификация продукции.

5. *Рекурсивные связи типа «один-к-одному».*

Связи представленные на рис. 1.30 невозможны. Тип связи, изображенный на рис. 1.31, применяется редко – для показа альтернативы.

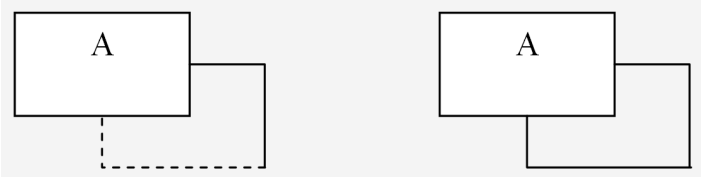


Рис.1.30. Неправильные рекурсивные связи типа «один-к-одному»

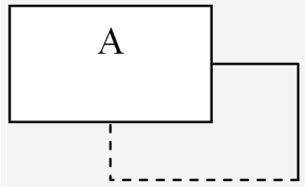


Рис.1.31. Рекурсивные связи типа «один-к-одному»

6. *Рекурсивные связи типа «многие-ко-многим».*

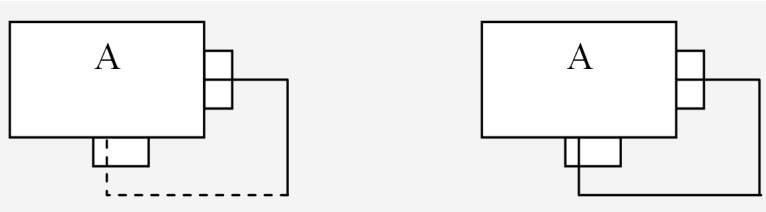


Рис.1.32. Неправильные рекурсивные связи типа «многие-ко-многим»

Связи, изображенные на рис. 1.32, невозможны.

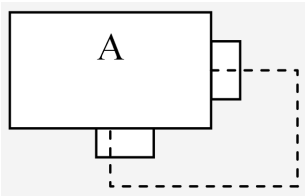


Рис.1.33. Допустимые рекурсивные связи типа «многие-ко-многим»

Тип связи, представленный на рис. 1.33, встречается и может быть прочитана для примера сущности «Компонент» следующим образом. Каждый Компонент может быть сборкой одного и более компонентов. Каждый Компонент может быть использован в одной и более других Компонентах.

Таким образом, анализируя полученную диаграмму, с точки зрения допустимости связи, можно сделать вывод о степени ее корректности.

Различают ER-диаграммы предварительные, которые описывают предметную область в общем виде и окончательные, как правило, в окончательных диаграммах связи «многие-ко-многим» не допускаются.

1.8.4. Проверка качества ER-модели

Автоматическая проверка качества – осуществляется CASE-средством по запросу пользователя. Проверке подлежат такие типы элемента ER-модели как, сущности, атрибуты, домены, связи, исключающие дуги.

Проверка осуществляется согласно списку параметров, подлежащих проверке. В результате выдаются грубые ошибки (Error) и не грубые (Massage).

При наличии грубых ошибок, генерация БД по ER-модели – невозможна.

Ошибки сущности:

Грубые:

- Имя не уникально в пределах проекта;
- У сущности нет ни одной связи;
- У сущности нет ключа
- Ключ состоит из необязательных элементов
- Сущность имеет только один подтип.
- Не грубые:
- У сущности нет описания
- Нет информации по объему сущности.

Ошибки атрибутов:

Грубые:

- Имя не уникально в пределах сущности
- Не определен формат и (или) длина.

Не грубые:

- Нет описания атрибутов.

Ошибки доменов:

Грубые:

- Имя не уникально в пределах проекта
- Не определен формат и (или) максимальная длина.

Не грубые:

- нет описания домена.

Ошибки связи:

- Грубые:
- Нет имени конца связи
- Невозможная связь
- Знак ключа на обоих концах связи
- Связь типа «многие-ко-многим»
- Незамкнутый конец связи.

Ошибки исключающих дуг:

Грубые:

- Дуги пресекают концы связи, имеющих:

Разную обязательность;

Разную множественность;

Разную ключевую идентификацию.

- Концы связей разных сущностей
- Только один конец связи
- Оба конца рекурсивной связи.

Для полной проверки качества ER-модели одного автоматического контроля недостаточно.

Всесторонняя проверка ER-модели должна проводиться экспертом. Основным экспертом экспертизы является список вопросов, раскрывающих наиболее распространенные ошибки.

Список вопросов для *экспертной проверки сущности*:

1. Имя каждой сущности – это существительное в единственном числе или компактное словосочетание.

2. Определяет ли имя сущности тип или класс объектов, а не отдельный экземпляр

3. отражен ли смысл каждой сущности в ее имени
4. является ли описание каждой сущности достаточно кратким и смысловым
5. если сущность имеет подтипы, то полностью ли их множество покрывает супертип - главная сущность
6. являются ли подтипы непересекающимися множествами
7. верно ли, что рассматриваемая сущность не является разновидностью другой сущности с упущенной рекурсивной связью
8. согласуется ли каждая сущность с принципами нормализации.
9. ключ действительно идентифицирует каждый экземпляр сущности
10. ключ сущности является минимальным.

Список вопросов для *экспертной проверки атрибутов*:

1. имя атрибута – существительное в единственном числе
2. отражен ли смысл каждого атрибута в его имени
3. является ли описание каждого атрибута достаточно кратеим и смысловым
4. атрибут не должен представлять упущенную связь, так ли это
5. атрибут не должен быть агрегатом других данных, то есть невозможно его разбиение на другие
6. если атрибут обязательный, всегда ли известно его значение
7. если определено множество значений атрибута, описан ли смысл каждого значения

Список вопросов для экспертной проверки связи

1. Данная связь действительно необходима
2. Если связь обязательная, то всегда ли определена сущность с другого конца.

Для *экспертной проверки доменов* необходимо ответить на вопрос: полностью ли домен описывает множество значений атрибута, их формат.

Рассмотрим пример ER-диаграммы для заказов (рис. 1.34).

Рис.1.34. Пример ER-диаграммы для заказов

1.8.5. Принципы генерации предварительного проекта БД

Генерация предварительного проекта БД предполагает следующие шаги:

1. *Шаг 1*: преобразование базовых конструкций ER-модели в предварительный проект БД. Каждая сущность супертип преобразуется в таблицу. При этом сущностью супертипом является та, которая не является подтипом.

2. *Шаг 2***:** каждый атрибут транслируется в колонку таблицы. Не обязательные атрибуты сущности супертипов, а так же все атрибуты сущности подтипов становятся не обязательными колонками. Обязательные атрибуты супертипа становятся обязательными колонками. Если сущность содержит подтипы, то генерируется колонка Тип_Имя сущности. Для получения полноценных колонок необходимо задать в ER-модели их тип и максимальную длину. Ключевые атрибуты порождают ключевые колонки.

3. *Шаг 3*: любая связь «один-к-одному» или «один-ко-многим» порождает дополнительные колонки, которые копируются из таблицы, соответствующей сущности на конце противоположном активному концу связи.

В качестве активных, то есть, участвующих в генерации концов связи, могут использоваться те, которые связаны с одним экземпляром сущности с другого конца (рис. 1.35).

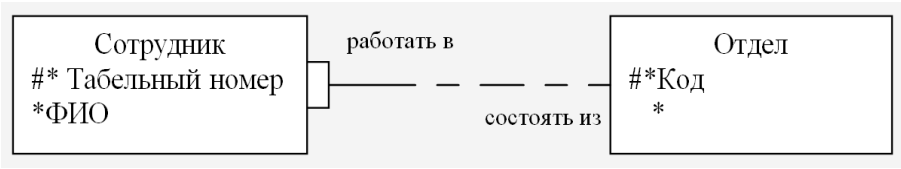


Рис. 1.35. Соединение сущностей «Сотрудник» и «Отдел»

Если связь со стороны активного конца связи ключевая, то она порождает ключевые колонки и не ключевые – в противном случае.

Таким образом, может возникнуть волна переходов ключевых атрибутов по сущности (рис.1.36).

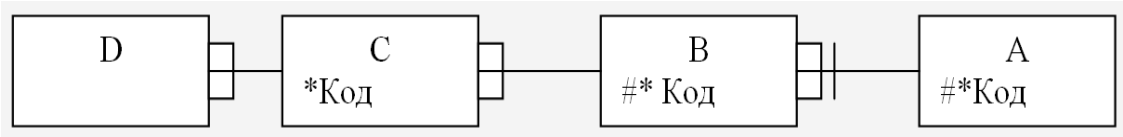


Рис. 1.36. Переход ключевых атрибутов по сущностям

Связи «многие-ко-многим» игнорируются. Особое внимание следует уделить генерации связи, пересеченной исключающей дугой. Такая связь может осуществляться двумя способами – с оптимизацией и без оптимизации.

Для генерации с оптимизацией необходимо выполнения условия полного совпадения формата и максимальной длины атрибутов альтернативной сущности (рис. 1.37).

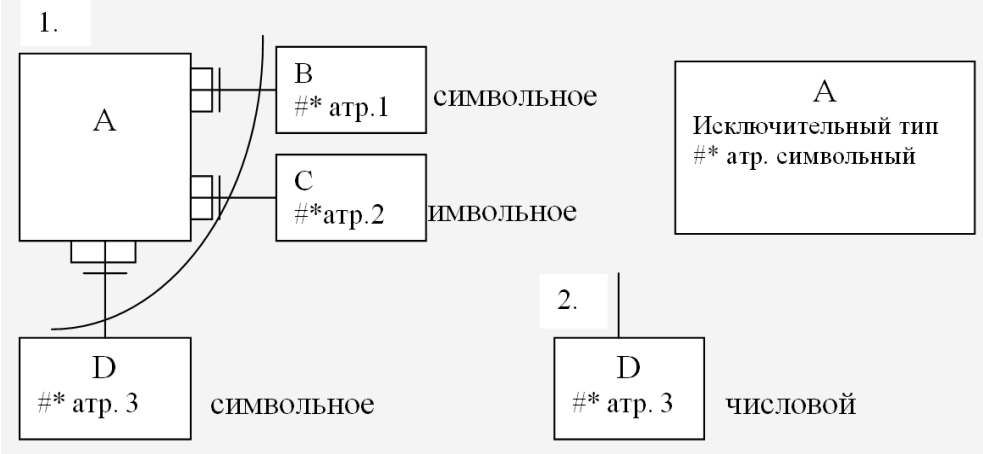


Рис.1.37. Генерация связей, пересеченных исключающей дугой

При генерации с оптимизацией будет создана колонка исключающий тип, предназначенная для того, чтобы различить связи, пересеченные исключающей дугой.

Оптимизация заключается в сокращении числа генерируемых колонок, особенно в случае, если большое количество атрибутов в альтернативной сущности.

Предположим, что атрибут сущности D имеет числовой формат. В этом случае генерация с оптимизацией невозможна, тогда получается таблица следующего вида (рис. 1.38):

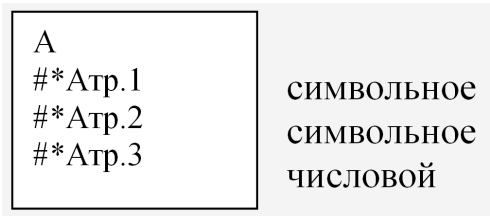


Рис.1.38. Оптимизация атрибута, имеющего числовой формат

Рассмотрим структуры данных для описания характеристик оборудования.

Задание: оборудование представлено совокупностью характеристик, причем каждый вид оборудования имеет разный набор параметра. Разработать структуру данных, описывающую оборудование.

Первое решение – представить информацию о параметрах в одной таблице (табл. 1.20). Число полей в таблице необходимо сделать таким, чтобы можно было описать максимальное число параметров.

Таблица 1.20

Описание характеристик оборудования

Код оборудования	Наименование	Название 1	Значение 1	Название 2	Значение 2
1	Телевизор Sony	диагональ	21	Тип экрана	1
2	Стиральная машина Indesit	Класс стирки	1	Количество, кг	5

Недостатки:

- Часть информации в таблице остается незаполненной;

- Могут возникнуть проблемы при осуществлении выборки данных, за счет того, что часть информации содержится вне таблицы.

Второе решение – представим информацию об оборудовании в разных таблицах.

Рис.1.39. Структура данных для описания характеристик оборудования

Для облегчения ввода значений можно создать вспомогательную таблицу, где описывается перечень характеристик для конкретного вида оборудования.

При вводе информации для выбранного вида оборудования формируется форма, затем информация из формы переписывается в таблицу «Характеристики оборудования» (рис. 1.39).

1.9. Особенности обработки БД в сетевом режиме

1. *Режим «Файл-сервер». Сервер* – это компьютер, устройство, ресурс, который может отдавать все свои ресурсы или часть другим пользователям. То, что потребляет сетевой ресурс, называется клиент.

Работа в сети организована с помощью топологий (шинная, звезда, кольцо).

При работе в режиме файл-сервера сетевое программное обеспечение скрывает от пользователя особенности конкретной сети, что приводит к тому, что пользователь работает с другим компьютером как с дополнительным диском (каталогом).

В режиме файл-сервера клиенту поставляются файлы.

Например, найти информацию в файле. Файл перекачивается с сервера на рабочую станцию, и поиск осуществляется на компьютере клиента.

Недостаток – при большом числе пользователей наступает перегрузка сетевого трафика.

2. *Система архитектуры «Клиент-сервер».* БД находится на сервере, клиенты работают с ней с помощью запросов. Для поиска информации в файле на сервер отправляется запрос, сервер осуществляет поиск, а по сети передается ответ, вследствие чего сетевой трафик разгружен, но возрастает нагрузка на сервер.

Для осуществления запросов в БД разработан специальный структурированный язык запросов SQL.

При переходе на архитектуру клиент-сервер, разработчики чаще всего ожидают повышения производительности. Следует отметить, что повышение скорости обработки данных не самая главная причина перехода к системе архитектуре клиент-сервер.

Хотя за счет уменьшения трафика при работе с большим числом станций – увеличивается производительность.

При небольших сетей с небольшим объемом данных и отсутствием реальной необходимости в их защите, возможно клиен-сервер не будет оптимальным, кроме того, серверы БД изначально создаются работы с действительно большими объемами данных и большим количеством пользователей.

Объем данных БД 100 Гб и более, для такой нагрузки СУБД класса Access не рассчитаны.

Серверу БД, как правило, требуются более мощная техника, по сравнению с той, что на рабочих станциях. При разработке структуры БД нужно стремиться, чтобы она была рационально спроектирована. Неудачно спроектированная БД не даст желаемой производительности.

1.10. Классификация информации

Основания классификации - признаки сходства или различия элементов.

Совокупность правил распределения оснований классификации образуют систему классификации.

Каждому объекту в системе классификации присваивается шифр (код) в соответствие с определенной системой кодирования. Существует четыре основные системы кодирования технико-экономической информации.

Две первые являются классификационными, так как построены на заранее определенной системе классификации, оставшиеся – регистрационные.

1. *Последовательная система кодирования*. Соответствует иерархической системе классификации. Шифр каждой нижестоящей группировки образуется путем прибавления цифр к вышестоящей.

Шифр обладает информативность, но громоздкий. В связи с чем, данная система кодирования используется в заранее разработанной системе классификации, например, в справочниках по народному хозяйству.

Цель общероссийских классификаторов информации – обеспечение обработки технико-экономической информации, получение статистических сведений с помощью средств вычислительной техники.

2. *Параллельная система кодирования.* Группировки цифр шифра не зависят друг от друга. Шифр имеет большую длину, и данная система применяется для целей маркировки, например, продукции.

3. *Порядковая система кодирования* – шифры присваиваются последовательно, так, чтобы уникально отличить один объект классификации от другого. Шифр компактен, но не информативен.

Данный вид классификации является основным при идентификации объектов в БД, на основе данного вида классификации чаще всего осуществляется связь между таблицами.

4. *Серийно-порядковая* – в отличие от порядковой шифры имеют диапазон по одному классификационному признаку.

Вопросы для повторения

1. Как классифицируются универсальные структуры данных?

2. Какие Вы знаете динамические структуры, которые могут изменять свой размер в зависимости от этапа выполнения программы?

3. Опишите основные этапы эволюции технологии обработки данных.

4. Что такое СУБД, и какие программные составляющие в себя включает?

5. Назовите известные Вам модели данных?

6. В чем состоит отличие иерархической модели данных от сетевой?

7. Какие основные операции допустимы над объектами в сетевой модели данных?

8. Какая модель данных имеет древовидную структуру данных?

9. Что такое «декартово произведение» и какая модель данных основана на этом понятии?

10. Что называется доменами и кортежами отношений?

11. Что собой представляет и когда используется составной ключ?

12. Перечислите основные операции обработки отношений реляционной модели данных.

13. Что такое нормализация отношений и как она применяется в проектировании реляционных баз данных?

14. Какие типы функциональных зависимостей Вы знаете?

15. Как называется отношение, у которого все атрибуты простые?

16. Как называется отношение, если оно находятся в 1НФ и каждый ключевой атрибут функционально полно зависит от составного ключа?

17. Какие неудобства порождает наличие транзитивных зависимостей?

18. Каково место процесса нормализации в проектировании реляционных баз данных?

19. Перечислите существующие виды отношений между таблицами при проектировании структуры баз данных?

20. Какие типы данных применяются в СУБД Access?

21. Что предполагает физическое упорядочивание данных в СУБД Access и к каким неудобствам приводит?

22. Какие виды индексов применяются для логического индексирования в СУБД Access?

23. Как в СУБД Access осуществляется обеспечение целостности данных?

24. Как называется логическая непротиворечивость данных одной и другой таблицы?

25. Как осуществить поиск, фильтрацию и сортировку в СУБД Access?

26. Назовите и охарактеризуйте уровни представлений, составляющие архитектуру информационной системы.

27. Перечислите основные требования, предъявляемые к проектированию баз данных.

28. Как сократить время поиска записей и обеспечить их упорядочение? Перечислите и опишите основные виды индексирования.

29. Опишите известные Вам способы удаления данных, их достоинства и недостатки.

30. Какими способами обеспечивается организация доступа к базам данных?

31. Как называется автоматизированный инжиниринг программных средств, то есть совокупность методик проектирования и сопровождения программных средств на всем из жизненном цикле, поддержанная взаимоувязанными средствами автоматизации?

32. Какие стадии жизненного цикла информационных систем поддерживают CASE-технологии?

33. Как называется метод разработки ER-модели, отражающей информационные потребности предприятия-заказчика и, не зависящей от физической реализации хранилищ данных и метода доступа к ним?

34. Что является графическим отображением ER-модели?

35. Из каких базовых элементов состоит ER-диаграмма?

36. По каким правилам осуществляется чтение ER-диаграммы?

37. Как происходит автоматическая проверка качества ER-модели?

38. На основании каких вопросов эксперт проводит экспертную оценку качества ER-модели?

39. Опишите основные шаги генерации предварительного проекта базы данных.

40. Как называется компьютер, устройство, которое может отдавать все свои ресурсы или часть другим пользователям (клиентам)?

41. Назовите известные Вам системы кодирования информации.

Резюме по теме

Значимость информационных ресурсов определяет то, что на их основе осуществляется управление другими ресурсами. Чтобы использование информационных ресурсов было эффективным, требуется разработка соответствующей технологии. Внедрение ЭВМ позволило перейти к автоматизации обработки данных. В настоящее время определяющим направлением организации и обработки файлов стала концепция баз данных. Широкое использование БД в управлении предприятием обусловлено оперативностью, гибкостью (получение ответов на любые запросы, наличие эффективных методов модификации данных и внесения изменений), безопасностью, целостностью, доступностью (возможность использования всей совокупности данных в БД для каждого пользователя).
Организация данных в СУБД может быть представлена различными моделями данных: сетевой, иерархической, реляционной. Из них наиболее распространена последняя.
Разработаны и применяются различные методы проектирования, разработки и повышения эффективности реляционных баз данных.

Тема 2. Базы данных в среде СУБД Access'2003

Цели и задачи изучения темы
Целью изучения темы является формирование устойчивого представления об основных инструментальных средствах и возможностях СУБД Access '2003, определение сферы ее применения и круга решаемых задач.

Задачи изучения темы:

- изучить факторы, влияющие на выбор СУБД;
- сформировать представление об основных понятиях СУБД Access '2003;
- рассмотреть возможности СУБД Access '2003;
- изучить степень реализации в MS Access функциональных потребностей пользователей разного уровня.

2.1. Создание БД в Microsoft Access

Базой данных в MS Access называется совокупность таблиц, форм, отчетов, запросов, модулей, макросов. Вся эта совокупность запоминается в одном файле базы данных формата mdb.

2.1.1. Создание новой базы данных

Если вы создаете новую базу данных, то надо после запуска Access выбрать позиции меню **Файл/Создать базу данных** и в появившемся окне **"Создание"** выбрать позицию **"База данных"** (рис. 2.1)

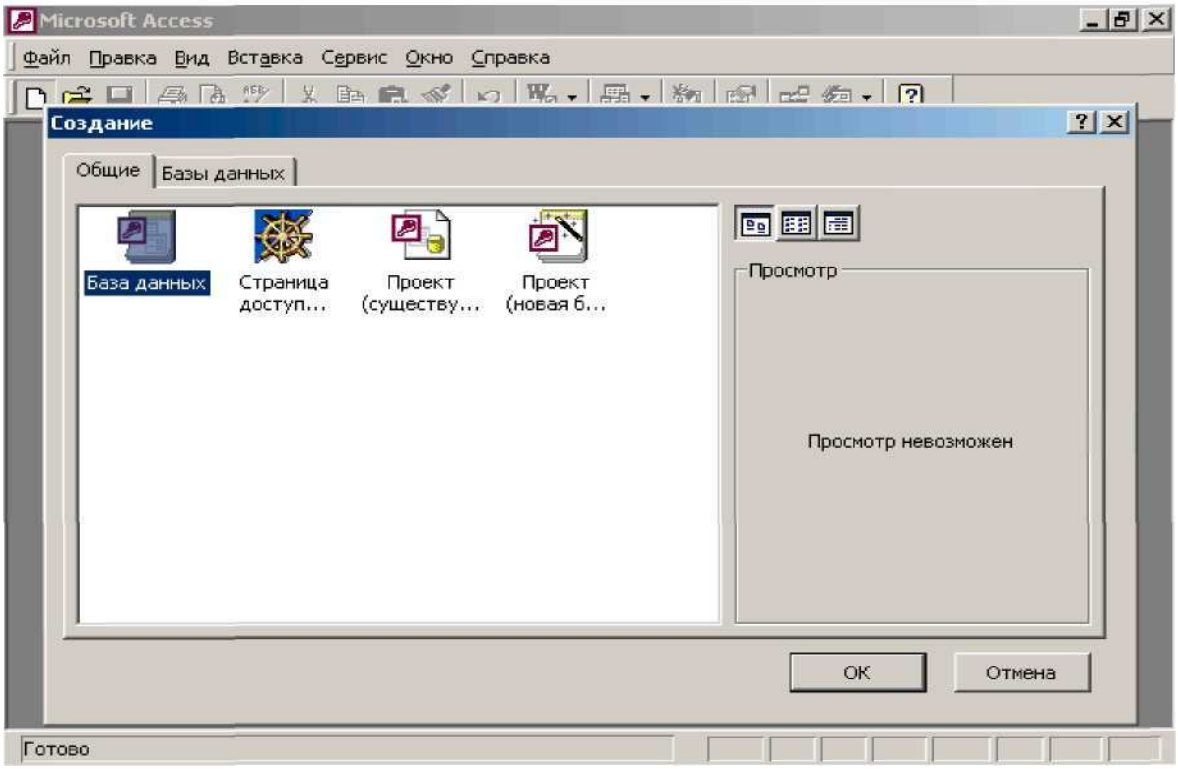


Рис. 2.1. Создание новой базы данных (экран 1)

В появившемся окне **"Файл новой базы данных"** (рис. 2.2) надо задать имя создаваемого файла БД и определить место, где он будет храниться, после чего, нажать кнопку "Создать". В нашем примере для файла базы данных задано имя «Демонстрационная» и он хранится в папке «Базы данных»

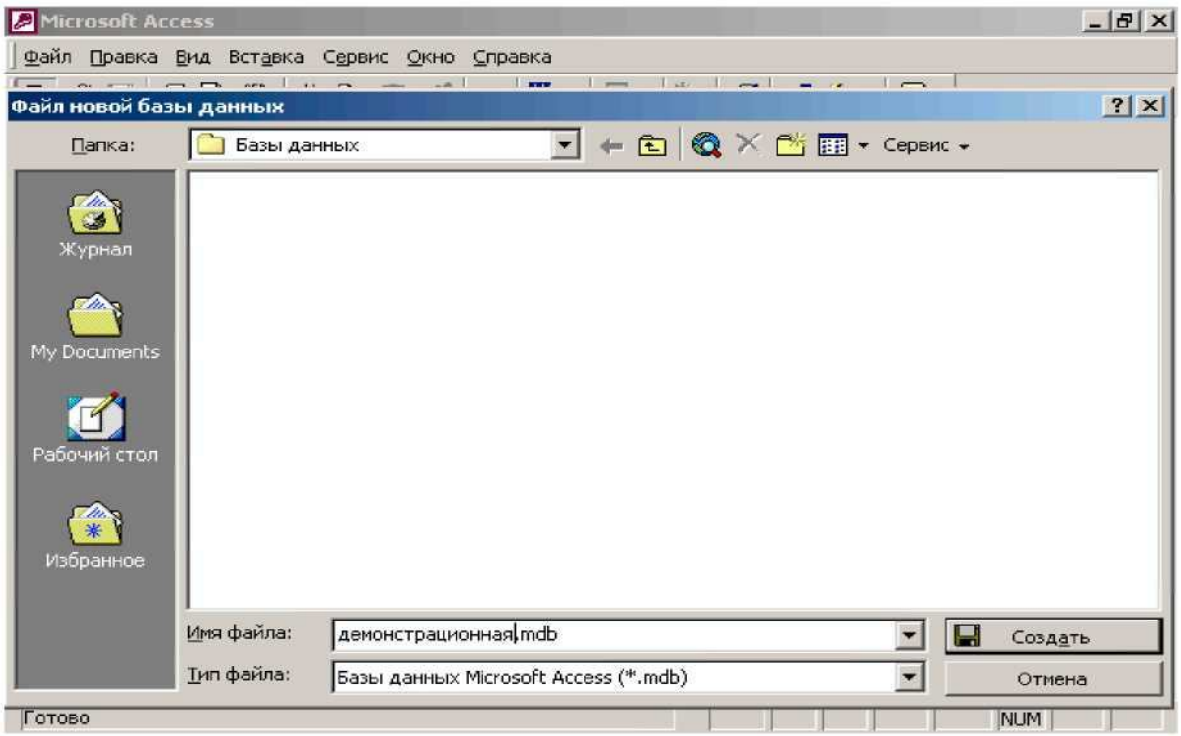


Рис. 2.2. Создание новой базы данных (экран 2 - задание имени базы данных)

После выполнения этих шагов появится экран "[название]: база данных" (рис. 2.3).

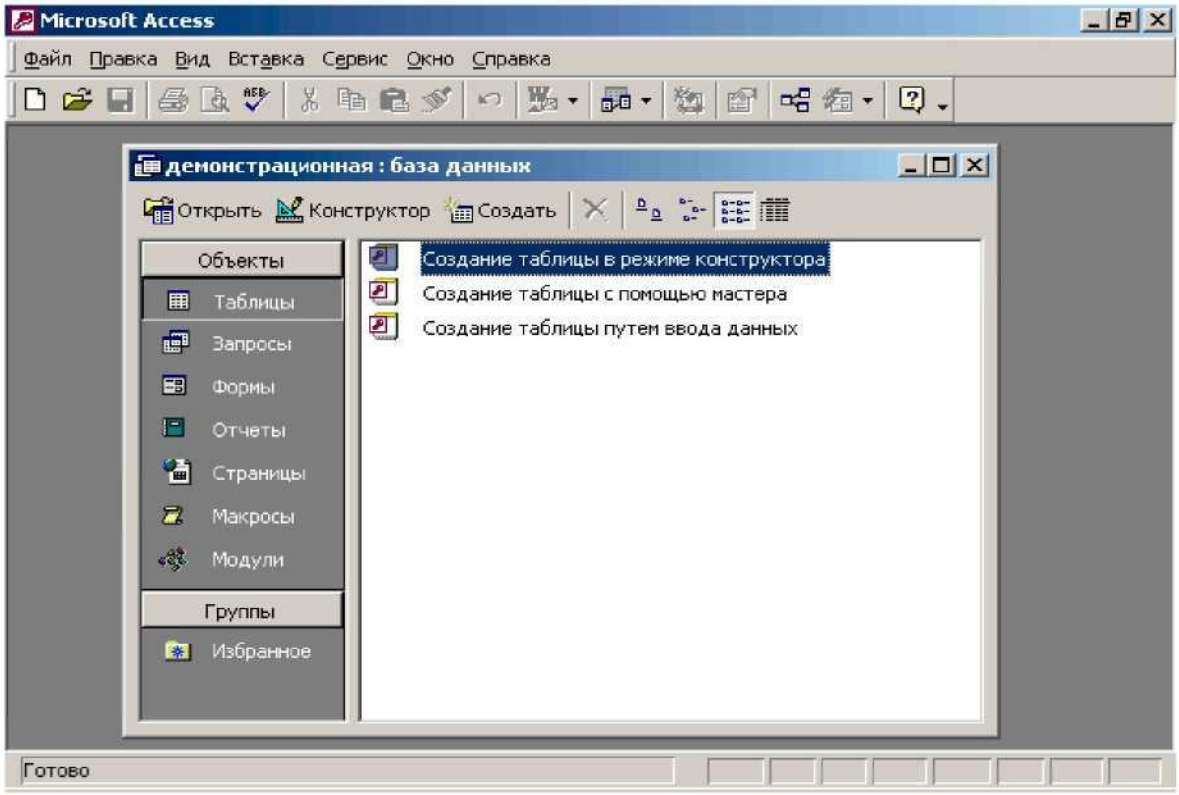


Рис. 2.3. Начальный вид окна базы данных

Далее в этой главе мы будем рассматривать только те вопросы, которые традиционно относятся к созданию баз данных, а именно создание таблиц и установление связей между ними.

2.1.2. Создание таблиц и связей между ними

Создание базы данных начинается с создания таблиц, в которых и хранится информация о предметной области. База данных обычно включает несколько взаимосвязанных таблиц. Для создания новой таблицы в окне **"Базы данных"** надо выбрать закладку **"Таблица"** и нажать кнопку **"Создать",** в результате чего появится окно **"Новая таблица".**

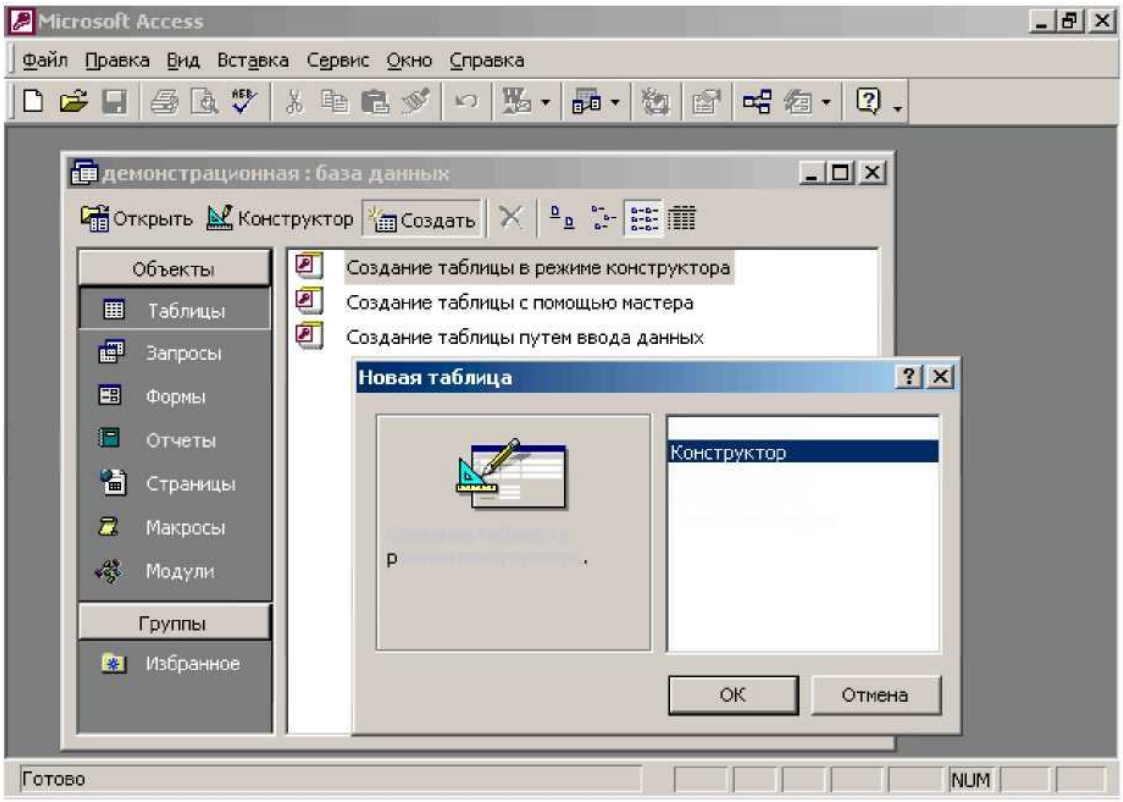


Рис. 2.4. Выбор способа создания таблицы

Создать таблицу можно в разных режимах: режиме таблицы, конструктора, мастера таблиц, импорта таблиц и связи с таблицами. Начнем рассмотрение возможностей создания таблиц с режима конструктора, как наиболее часто используемого. Для этого в появившемся окне выберите режим создания нового объекта - **"Конструктор".**

После чего появится окно для описания структуры таблицы и других ее характеристик (рис. 2.5).

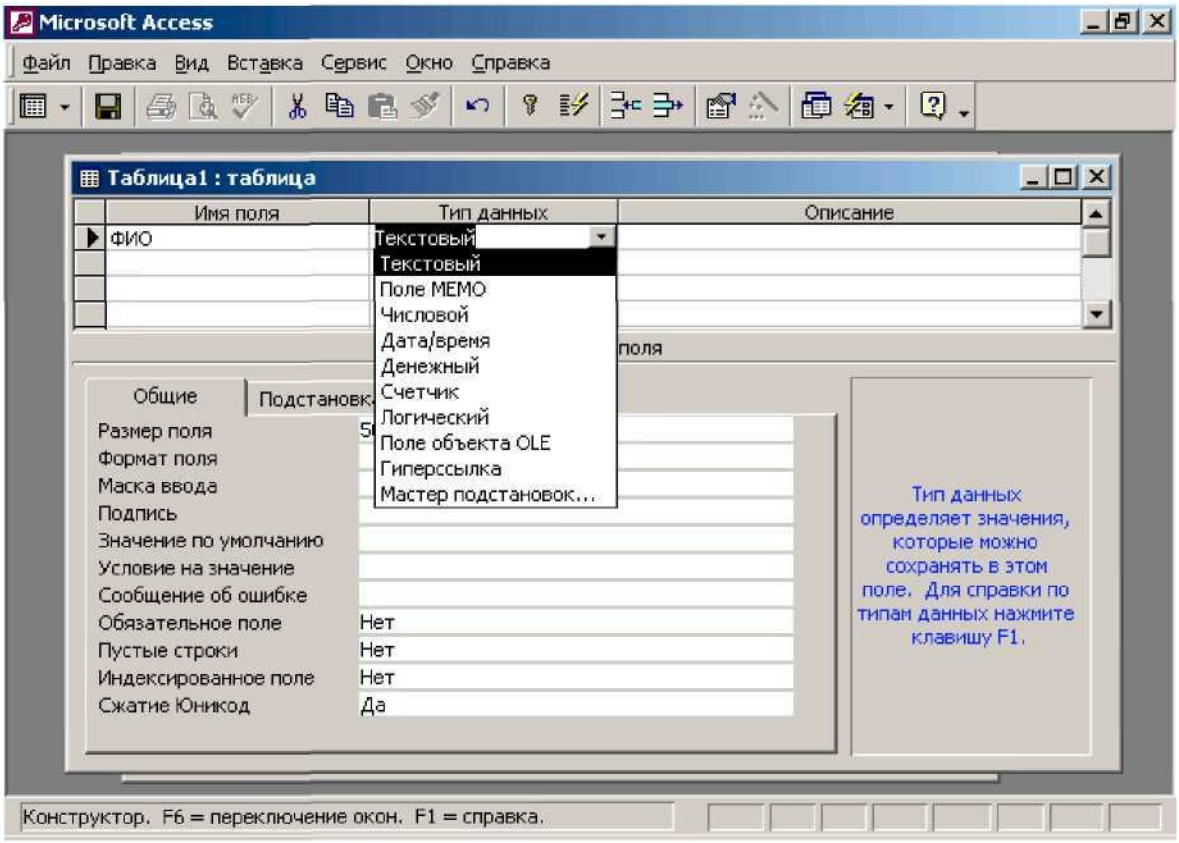


Рис 2.5. Описание структуры таблицы и других ее характеристик

В табличной форме надо последовательно описать все поля создаваемой таблицы. Сначала задается имя поля. Access допускает задание длинных имен с пробелами на русском языке.

В Microsoft Access действуют следующие ограничения на имена полей:

- имя должно содержать не более 64 символов;
- имя может включать любую комбинацию букв, цифр, про белов и специальных символов за исключением точки (.), восклицательного знака (!), надстрочного символа (`) и прямых скобок ([ ]);
- имя не должно начинаться с символа пробела;
- имя не должно включать управляющие символы (с кодами ASCII от 0 до 31).

Хотя пробелы внутри имен полей и являются допустимыми, они могут при некоторых обстоятельствах вызывать конфликты при работе с другими системами. Поэтому их не рекомендуется использовать. Вообще к заданию длинных имен на русском языке надо относиться с осторожностью, особенно, если есть вероятность, что создаваемое приложение будет в дальнейшем использоваться в распределенных гетерогенных системах.

При задании имен не допускайте их совпадения с зарезервированными словами. Например, не следует давать полю имя Count, Name и т. п.

Имя поля должно быть уникальным в пределах таблицы. И хотя система не запрещает использование одинаковых имен полей в разных таблицах, избегайте

использования одинаковых имен для обозначения разных по смыслу атрибутов. Имя должно быть понятно не только в контексте данной конкретной таблицы. Так, например, если в таблице "СОТРУДНИК" есть поле "Код", и такое же поле есть в таблице "КАФЕДРА", то в первом случае это будет код сотрудника, а во втором -код кафедры. Многие системы (и Access в том числе) автоматически связывают таблицы по полям, которые имеют одинаковые имя, тип и длину. Если имена даны непродуманно, то могут либо возникнуть неправильные связи, либо процесс задания связей будет несколько сложнее, чем при правильном задании имен.

После задания имени надо выбрать тип поля. Если щелкнуть мышкой по свободной ячейке графы **"Тип поля",** то высветится список допустимых типов полей (см. Рис. 2.5), из которого и следует выбрать подходящий для описываемого поля тип. Имя и тип поля должны задаваться обязательно. Графа **"Описание"** может не заполняться. Эта графа используется в целях документирования проекта.

Допустимые типы полей в Access-2003 и их краткая характеристика приведены в таблице 2.1.

В представленном материале длинные имена с пробелами даются исключительно с целью достижения большей наглядности излагаемого материала.

Таблица 2.1.

Допустимые типы полей в Access2003

Тип данных	**Содержимое поля**	**Размер**
Текстовый	Текст или числа, не требующие проведения расчетов, например, номера телефонов, коды и т. п...	Максимальное число символов -255.
Поле MEMO	Длинный текст или комбинация текста и чисел.	До 65535 символов.
Числовой	Числовые данные, используемые для проведения расчетов.	1, 2, 4 или 8 байт
Дата/время	Даты и время, относящиеся к годам с 100 по 9999 включительно.	8 байт.
Денежный	Специальный формат для представления числовых данных. Точность - до 15 знаков в целой и до 4 знаков в дробной части.	8 байт
Счетчик	Уникальные последовательно возрастающие (на 1) или случайные числа, автоматически вводящиеся при добавлении каждой новой записи в таблицу.	4 байт
Логический	Поля, которые могут содержать одно из двух возможных значений (True/False, Да/Нет).	1 бит

Поле объекта OLE	Объект, связанный или внедренный в таблицу Microsoft Access.	До 1 Гбайт (ограничивается объемом диска).
Гиперссылка	Строка, состоящая из букв и цифр, и представляющая адрес гиперссылки. Адрес гиперссылки может состоять максимум из трех частей *:текст* - текст, выводимый в поле или в элементе управления;адрес- путь к файлу (в формате пути UNC) или странице (адрес URL) *дополнительный адрес* - смещение внутри файла или страницы.	Каждая из трех частей в типе Гиперссылка может содержать до 2048 символов.

В списке допустимых типов полей (см. Рис. 1.5) имеется строка **"Мастер подстановок".** При его использовании можно создать поле, содержание которого формируется путем выбора значений из списка, содержащего набор постоянных значений или значений из другой таблицы/запроса. Если источником для подстановки выбран столбец другой таблицы, то тип и длина поля, созданного таким способом, будет определяться типом и длиной элементов, служащих источником для подстановки значений.

Выбор типа поля является важным шагом при проектировании БД. Принятое решение оказывает влияние на выполняемый при вводе контроль правильности данных, на допустимые операции над данными и особенности их выполнения, требуемый объем памяти, скорость выполнения операций, совместимость разных частей БД при работе в гетерогенной среде.

Предположим, что мы создаем таблицу, содержащую сведения о профессорско-преподавательском составе. Состав и тип полей создаваемой таблицы представлены на рис. 2.6.

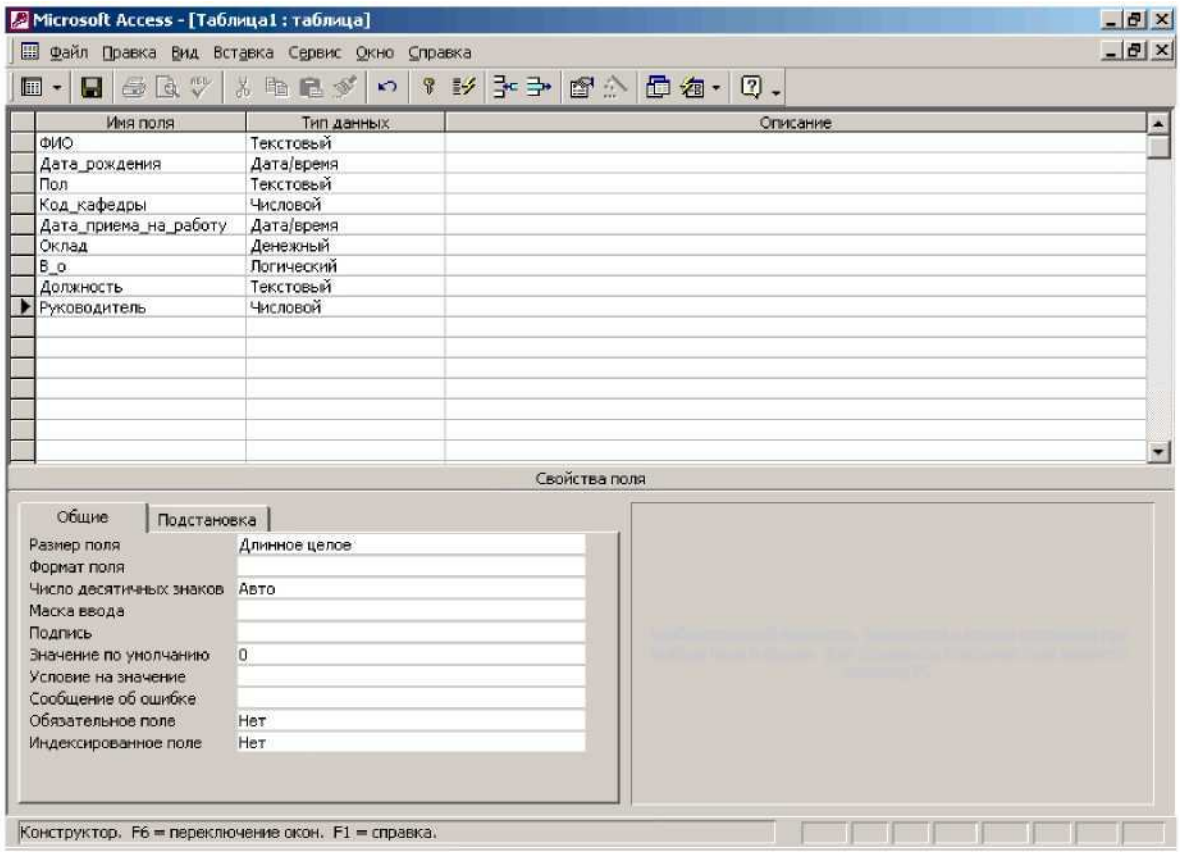


Рис. 2.6. Состав полей таблицы "СОТРУДНИК"

Введите описания всех полей.

Обратим внимание на поле «Должность». Для выбранной категории сотрудников имеется всего четыре возможные должности: ассистент, старший преподаватель, доцент и профессор. Хорошо было бы заменить ввод этих значений выбором их из списка. В ранних версиях Access задавать домен (либо путем прямого ввода списка значений, либо путем связи с файлом подстановки) можно было только при создании запроса или экранной формы. В последних версиях стало возможным задать его и при описании таблицы.

Используем **"Мастер подстановок"** при определении типа данных поля "Должность".

Для этого можно либо при выборе типа указать **"Мастер подстановок"** (см. последнюю строку в ниспадающем списке типов полей на рис. 2.5), либо выбрать позицию "Поле **подстановки"** в меню **"Вставка".**

Последовательность шагов при создании поля подстановки изображены на рис 2.7-2.9. При создании поля с помощью мастера подстановок имя поля можно не задавать, а сразу перейти к столбцу **"Тип** данных" и выбрать в списке строку **"Мастер подстановок".** Имя поля будет задано позже в процессе создания поля с помощью мастера.

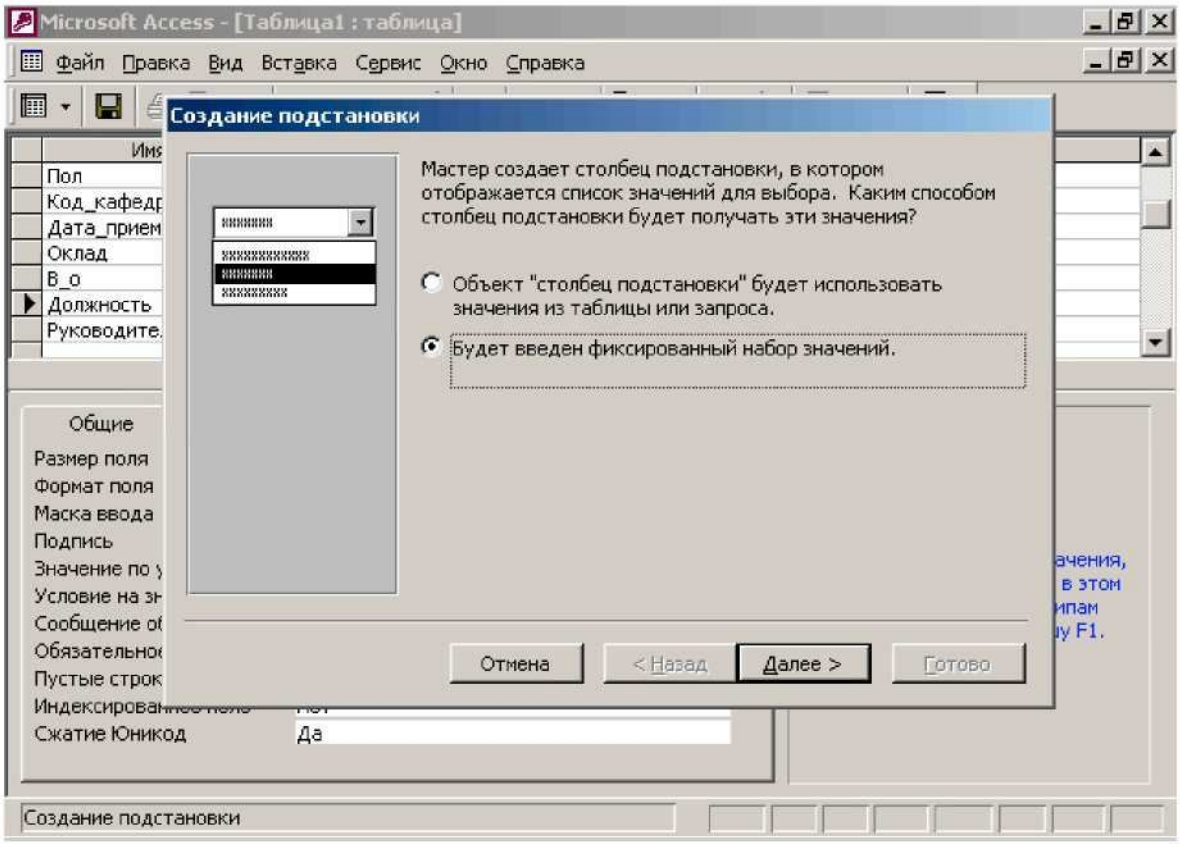


Рис. 2.7. Создание столбца подстановки. Начальный экран

Так как список создаваемый в рассматриваемом случае короткий и стабильный, то создадим столбец подстановки с фиксированным набором значений (рис. 2.7). В появившемся далее окне введем требуемые значения (рис. 2.8).

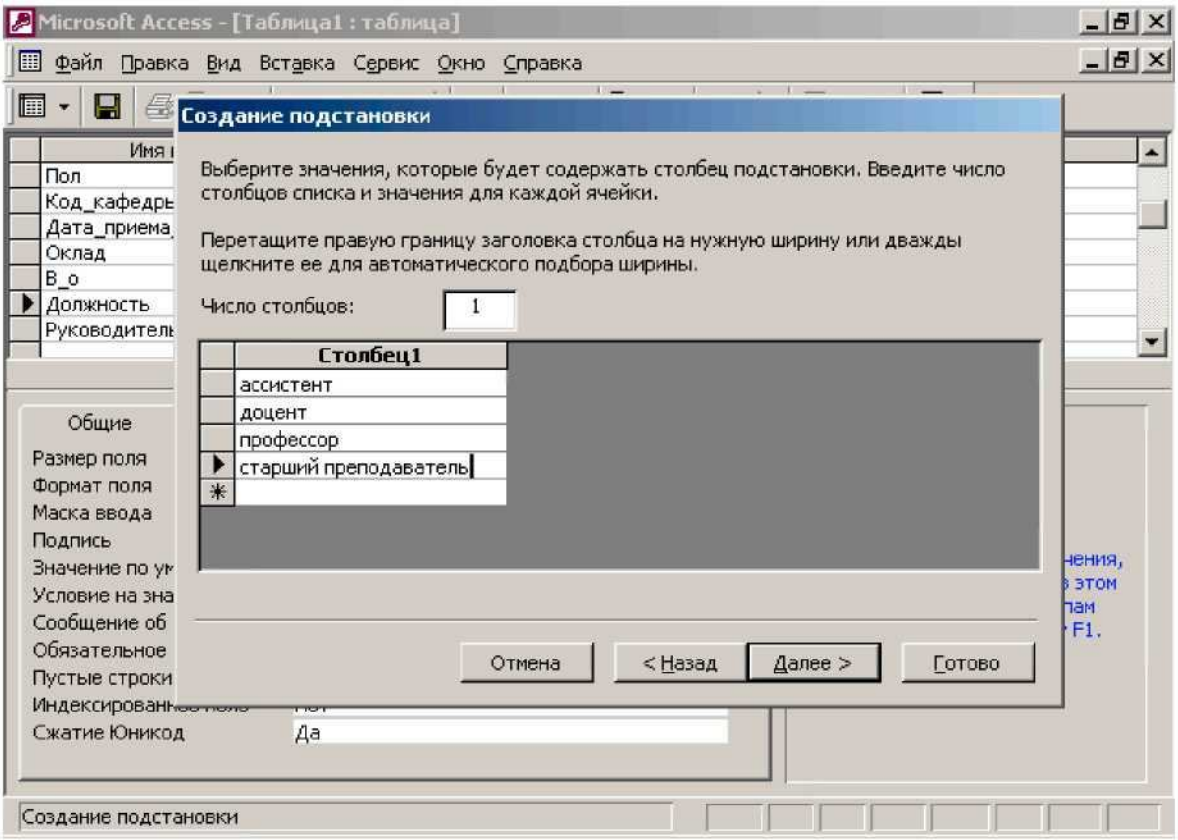


Рис. 2.8. Создание столбца подстановки. Столбец с введенным списком значений.

Далее зададим имя этого поля (рис. 2.9).

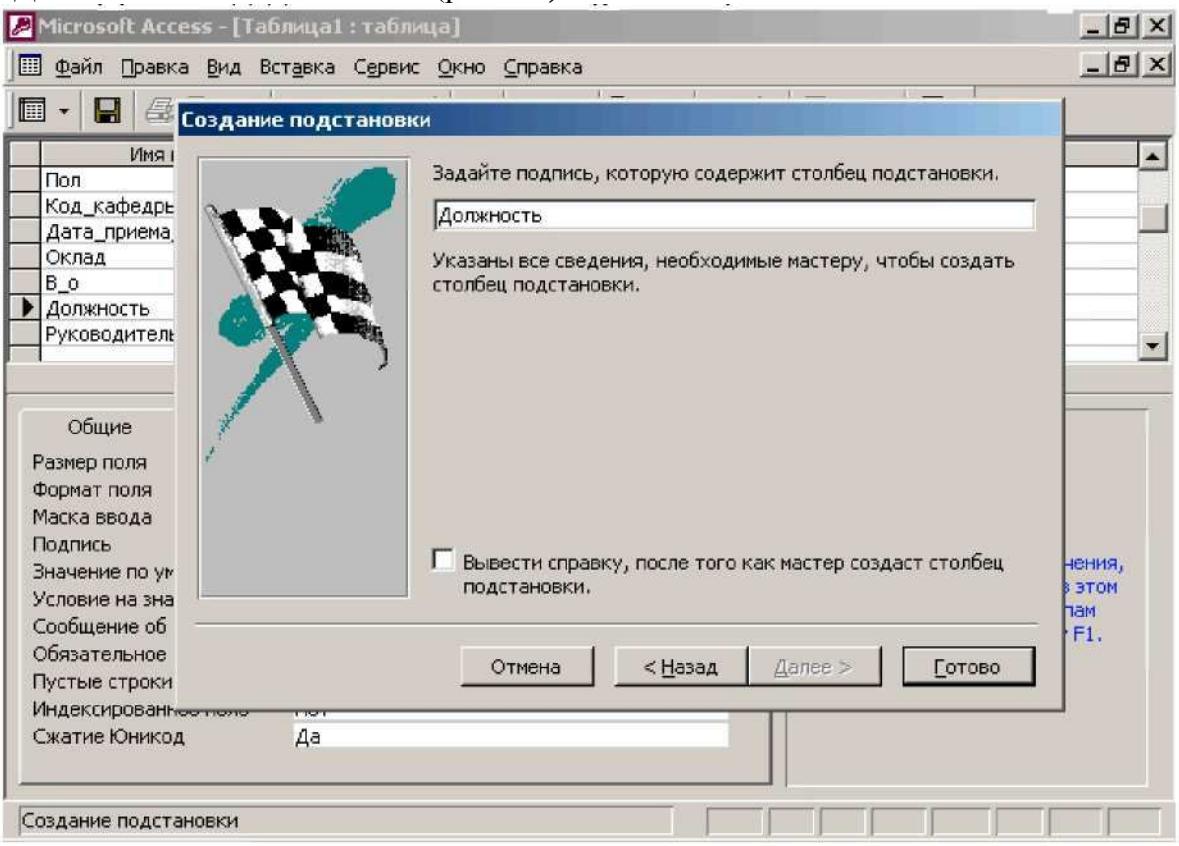


Рис. 2.9. Создание столбца подстановки. Задание имени столбца

При создании поля таким способом его тип будет "текстовый" и длина - 50. После создания поля с использованием мастера подстановок с фиксированным набором значений его тип и длину можно скорректировать.

При вводе данных в таблицу значения полей подстановки можно не вводить с клавиатуры, а выбирать из заданного списка. Чтобы нельзя было ввести значения, отсутствующие в списке, надо в свойствах поля на вкладке **"Подстановка"** в позиции **"Ограничиться списком"** задать значение "Да". В этом случае использование поля подстановки обеспечит не только более эффективный ввод данных, но и более жесткий контроль целостности базы данных.

Если число значений поля подстановки достаточно велико, и они могут меняться со временем, то следует использовать вторую альтернативу - использовать значения из другой таблицы/запроса (рис. 2.10-2.12). Эта возможность используется в нашем примере для поля «КОД_КАФЕДРЫ», значения которого будут браться из таблицы «КАФЕДРА». Естественно, что таблица «КАФЕДРА» должна быть предварительно создана.

Для создания поля подстановки, источником для которого служит другая таблица, лучше сначала создавать основную таблицу (в паре "КАФЕДРА"-"СОТРУДНИК" основной будет таблица "КАФЕДРА"), а затем создавать поле подстановки (в нашем случае это поле "Код_кафедры" в таблице "СОТРУДНИК"). Если поле уже было создано (как в нашем случае), то его можно скорректировать, выбрав в столбце **"Тип данных"** позицию **"Мастер подстановки"** (рис. 2.10) и далее выполнить те шаги, которые описаны ниже/

Если вы изучаете материал, выполняя на компьютере описываемые действия, то временно отложите выполнение шагов по созданию поля подстановки из другой таблицы. Завершите создание таблицы СОТРУДНИК как описано далее; затем создайте таблицу КАФЕДРЫ, свяжите эти таблицы. После этого откройте таблицу СОТРУДНИК в режиме «конструктор» и выполните описанные здесь шаги.

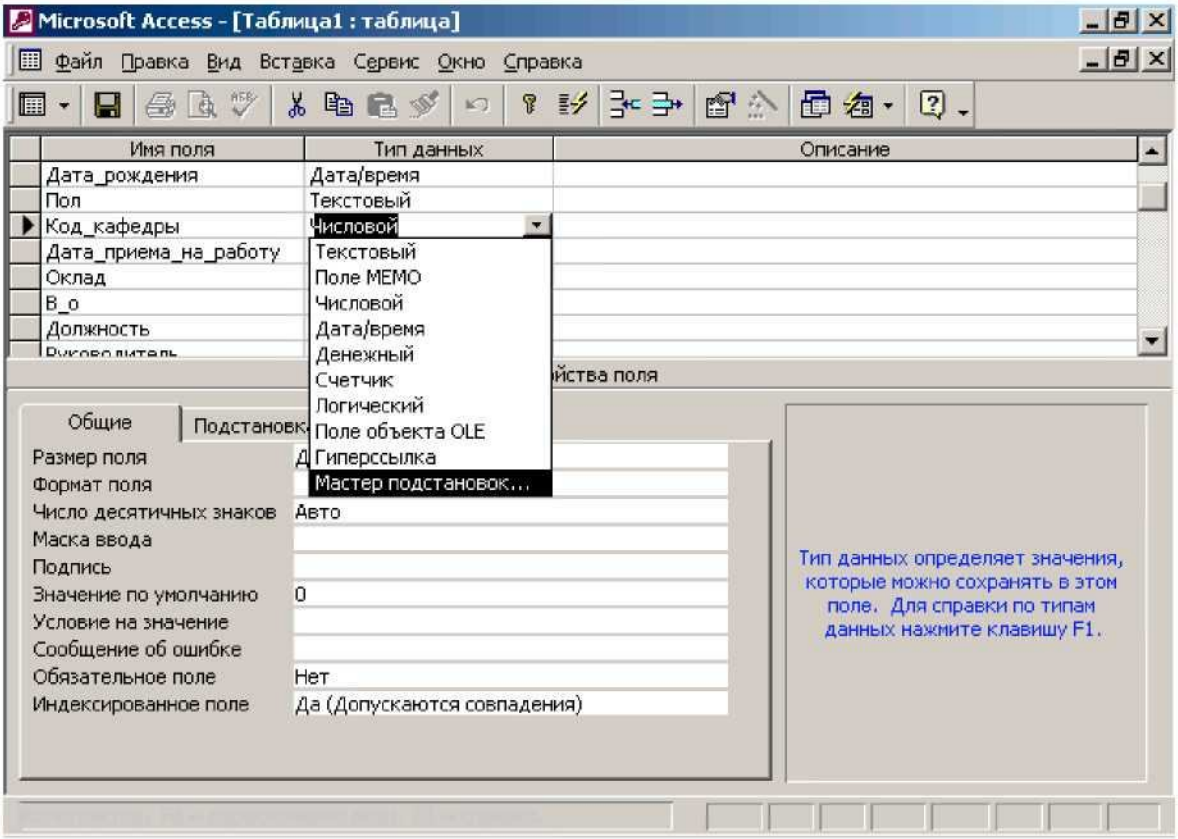


Рис. 2.10. Создание поля подстановки (шаг 1)

В окне **«Создание подстановки»** выбираем альтернативу **«Столбец подстановки» использует значения из таблицы или запроса»** (рис. 2.11).

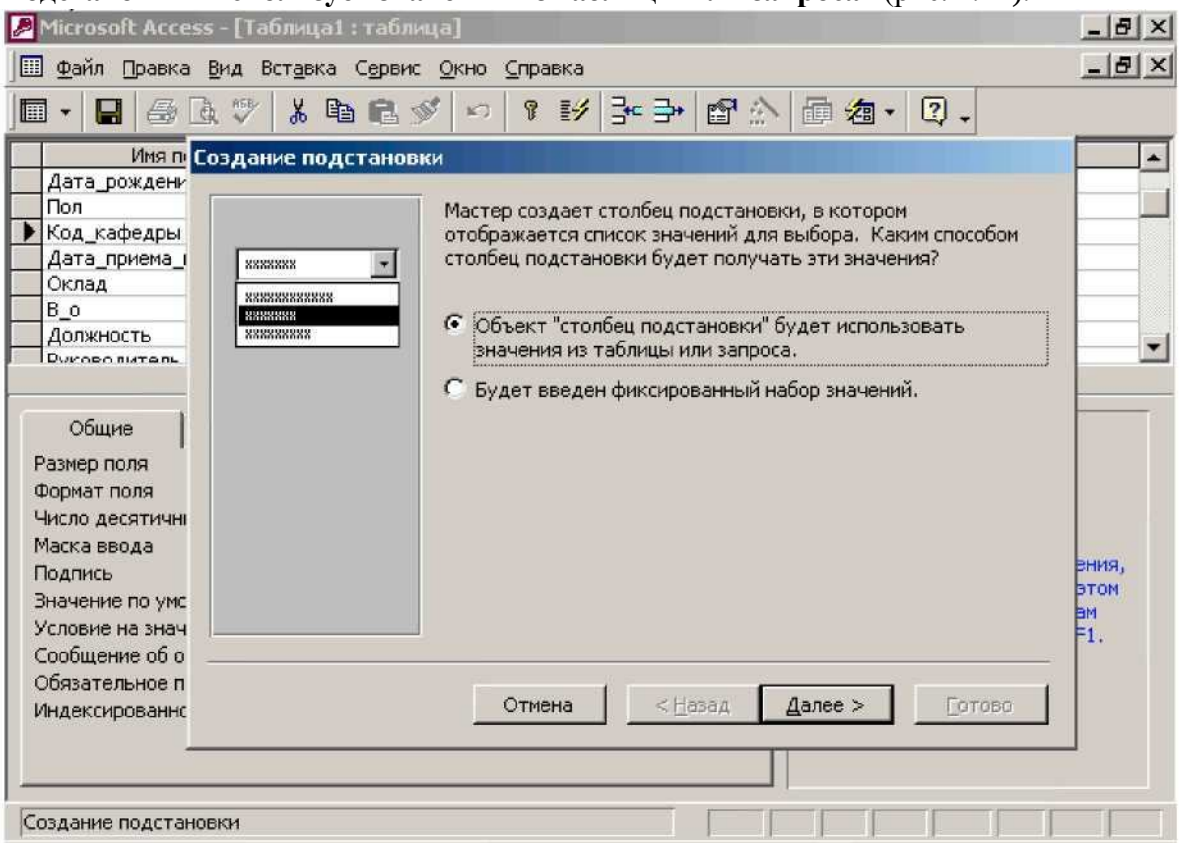


Рис. 2.11. Создание поля подстановки (шаг 2)

После этого надо выбрать таблицу/запрос, которая будет являться источником данных для описываемого поля (рис. 2.12)

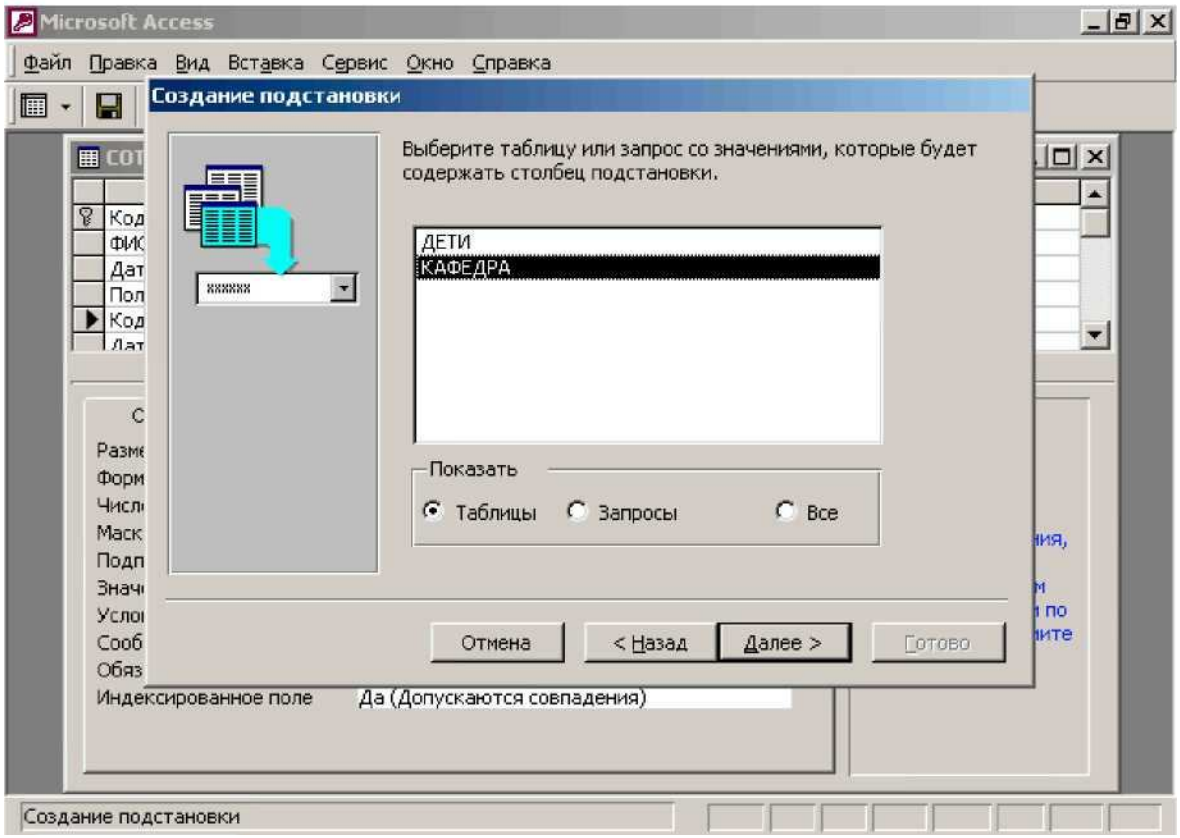


Рис. 2.12. Выбор таблицы/запроса источника для поля подстановки

Далее надо определить колонку таблицы-источника, значения из которой будут подставляться в описываемую колонку (рис. 2.13). В нашем примере таким полем является «КОД_КАФЕДРЫ». Но так как пользователь вряд ли помнит коды, то желательно, чтобы при выборе нужного значения высвечивались названия кафедр. Для этого в окно **«Выбранные поля»** следует перенести еще и поле «НАИМЕНОВАНИЕ_КАФЕДРЫ_ПОЛНОЕ» или «НАИМЕНОВАНИЕ_КАФЕДРЫ_КРАТКОЕ».

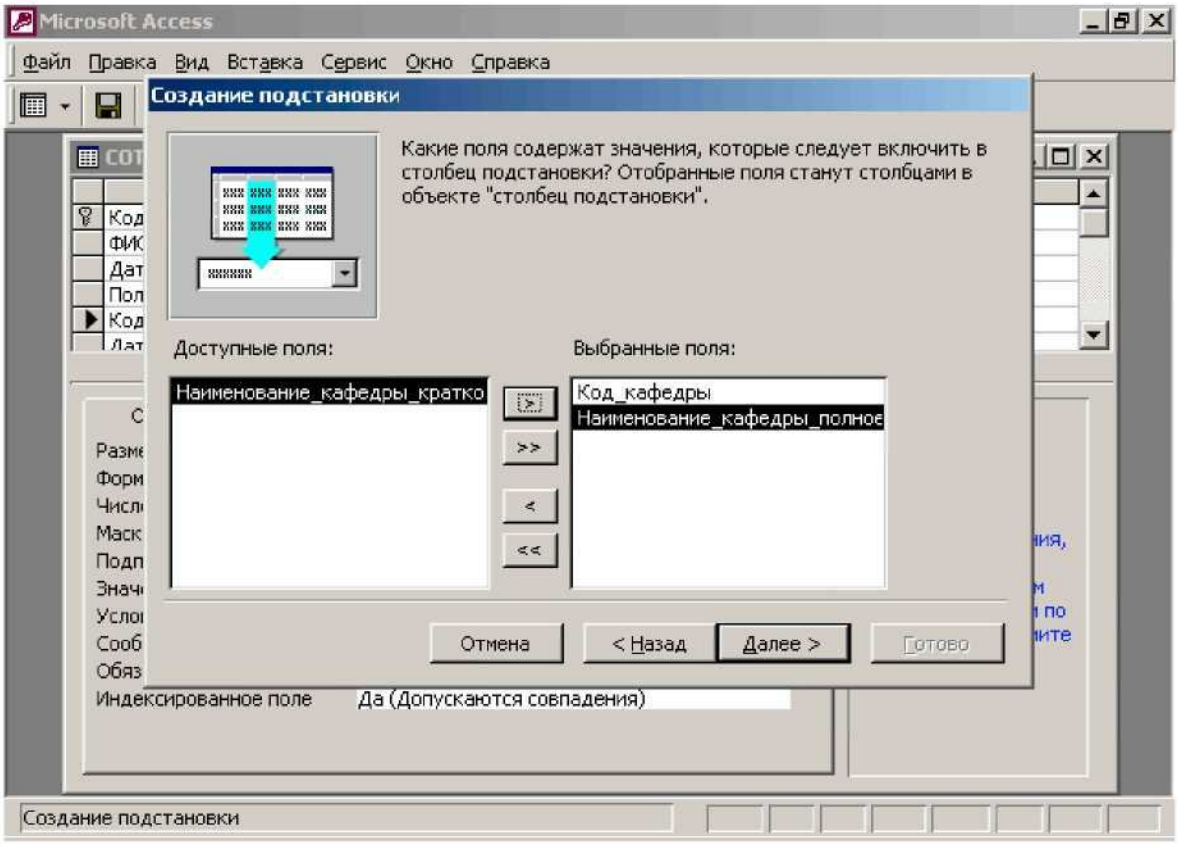


Рис. 2.13. Выбор колонки-источника для поля подстановки

В появившемся далее окне (рис. 2.14) можно не только задать ширину столбцов (позиционировавшись на границу столбца и перетащив ее в нужном направлении), но и определить, сколько столбцов будет выводиться на экран при вводе значения в это поле: если оставить знак «√» в позиции **«Скрыть ключевой столбец»,** то в нашем примере будет выводиться только «НАИМЕНОВАНИЕ_КАФЕДРЫ_ПОЛНОЕ». Если эту «галочку» убрать, то будут выводиться оба поля: «КОД_КАФЕДРЫ» и «НАИМЕНОВАНИЕ_КАФЕДРЫ_ПОЛНОЕ».

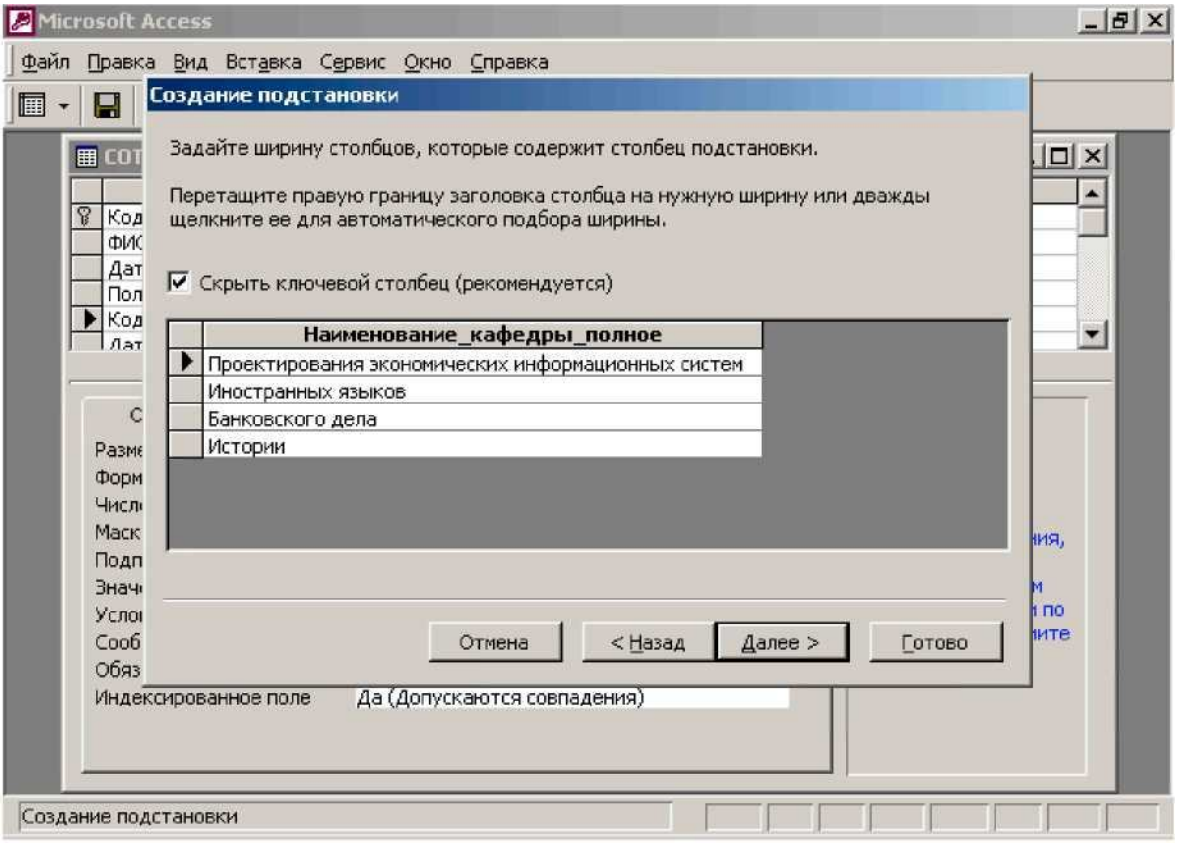


Рис. 2.14. Создание поля подстановки. Задание ширины столбцов

Определение ключа таблицы

Каждая реляционная таблица по определению имеет ключ. Access позволяет задавать ключ при описании таблицы, но также разрешает и отказаться от этой возможности. По ключу система автоматически выполняет индексирование, а также проверяет уникальность значений ключа при вводе новых записей или их корректировке.

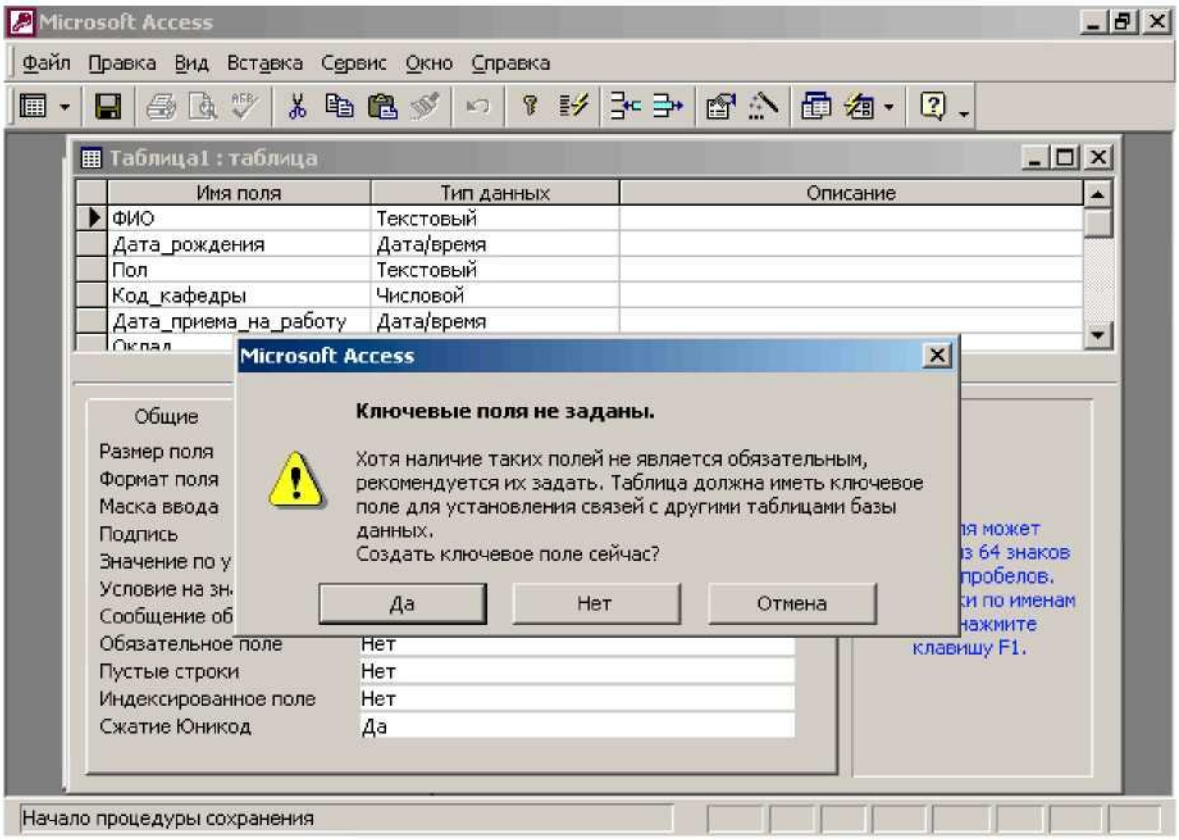


Рис. 2.15. Задание ключа таблицы в виде поля типа счетчик при закрытии таблицы после описания ее полей

Если Вы собираетесь в качестве ключа выбрать автоматически задаваемый системой код (т. е. поле типа "счетчик"), то можно это поле первоначально не описывать, а подтвердить необходимость его создания при завершении описания таблицы. Access создаст это поле автоматически (рис. 2.16).

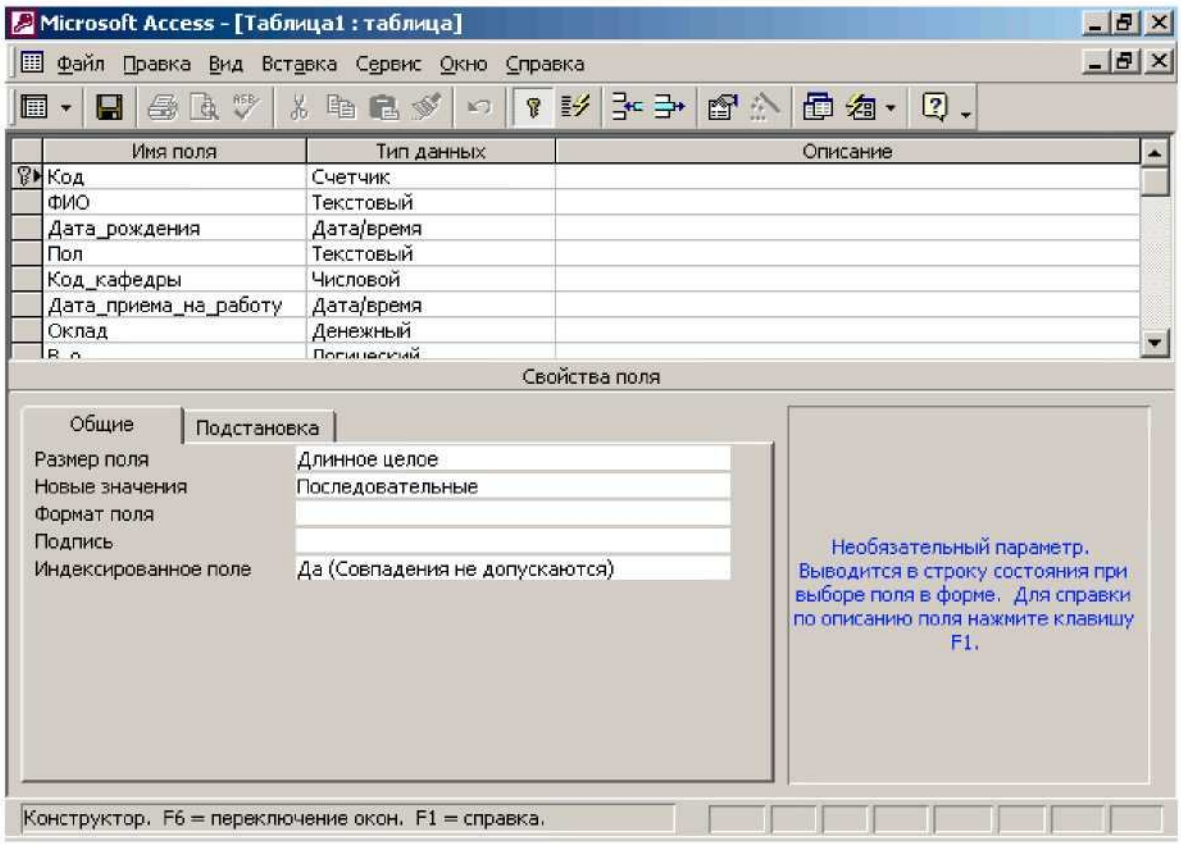


Рис. 2.16. Описание структуры таблицы с автоматически введенным системой ключевым полем Код

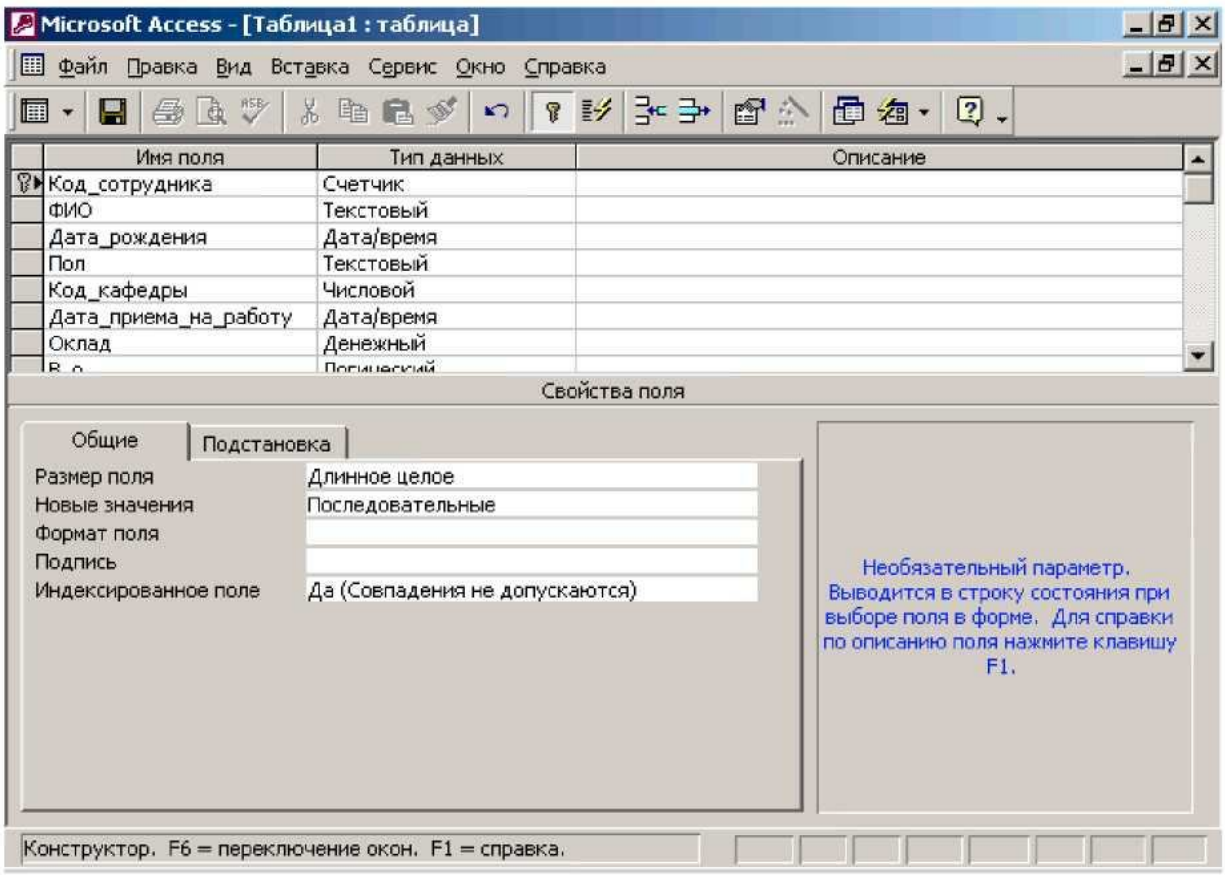


Рис. 2.17. Описание структуры таблицы с измененным ключевым полем

Как известно, ключ может быть составным. Чтобы определить составной ключ надо "отметить" соответствующую совокупность полей, а затем выполнить те же действия, что и при задании простого ключа.

Свойства полей

В нижней части экрана описания таблицы отображается список свойств выбранного поля. Перечень свойств будет зависеть от выбранного тапа поля (табл. 2.2).

Если вы определяете ключ самостоятельно, то это можно сделать несколькими путями: позиционироваться на соответствующее поле и нажать кнопу **"Ключевое поле"**, имеющую пиктограмму в виде изображения ключа, либо выбрать позицию меню **Правка/Ключевое поле,** либо воспользоваться правой кнопкой мыши для вызова контекстного меню, предварительно позиционировавшись на то поле, которое определяется как ключевое.

Для удобства дальнейшей работы с таблицей "СОТРУДНИК" зададим ключевое поле "КОД_СОТРУДНИКА" как показано на рис. 2.17.

Таблица 2.1

Свойства полей (в зависимости от типа поля)

Тип поля / Свойство	Текстовое	Логическое	МЕМО	Числовое	Дата/время	Денежный	Счетчик
размер поля	+			+		+	+
Число десятичных знаков				+		+	
Формат поля	+	+	+	+	+	+	+
Маска ввода	+			+	+	+	
Подпись поля	+	+	+	+	+	+	+
Значение по умолчанию	+	+	+	+	+	+	
Условие на значение	+	+	+	+	+	+	
Сообщение об ошибке	+	+	+	+	+	+	
Обязательное поле	+	+	+	+	+	+	
Пустые строки	+		+		+		
Индексированное поле	+	+		+	+	+	+

Новые значения							

Набор допустимых свойств вызывает некоторое удивление. Наверное, не всеми возможностями надо пользоваться. Так, обычно не рекомендуется проводить индексирование по логическому полю. Назначение поля МЕМО - хранение длинных текстов. Как и зачем задавать для них условия на значение - не совсем понятно. То же (но несколько в меньшей степени) относится и к формату поля данного типа, а также формату поля счетчик.

Некоторые из свойств полей понятны без дополнительных пояснений. Некоторые будут пояснены ниже на примерах, другие свойства будут пояснены позже при рассмотрении соответствующих тем.

Свойство **"Индексированное поле"** определяет, надо ли создавать индекс по этому полю. Индекс ускоряет выполнение запросов, в которых используются индексированные поля, операции сортировки и группировки. Свойство **"Индексированное поле"** может иметь следующие значения (таблица 2.3).

Таблица 2.2

Значения свойства «Индексированное поле»

Значения	**Описание**
Нет	(Значение по умолчанию). Индекс не создается.
Да (Допускаются совпадения)	В индексе допускаются повторяющиеся значения.
Да (Совпадения не допускаются)	Повторяющиеся значения в индексе не допускаются.

В рассматриваемом нами примере в связи с тем, что по полю ФИО часто осуществляется поиск и осуществляется упорядочение информации, желательно по нему произвести индексацию. Так как среди сотрудников возможны однофамильцы, то должны быть разрешены совпадения значений индексируемого поля (рис. 2.18).

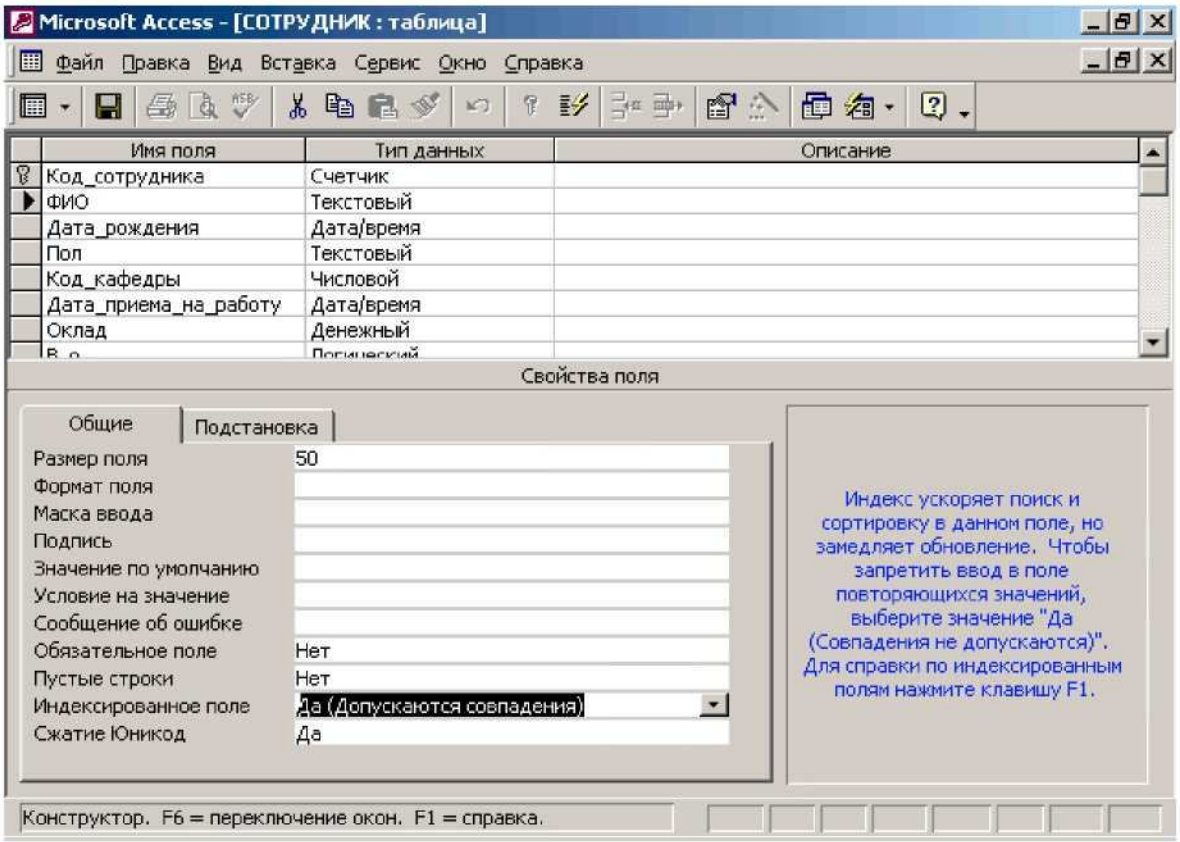


Рис. 2.18. Индексирование. Задание обязательности значений

Не допускается создание индексов для полей MEMO, гиперссылок и объектов OLE.

Ключевое поле "КОД", созданное системой автоматически, имеет тип **«Счетчик»** (рис. 2.10). Только для полей этого типа имеется свойство **«Новые значения».** Оно определяет способ увеличения значения поля счетчика при добавлении в таблицу новых записей.

Свойство **«Новые значения»** может иметь следующие значения:

- последовательные - значение поля счетчика увеличивается на 1 в каждой новой записи;

- случайные - поле счетчика в новой записи получает случайное значение типа Long Integer.

Свойство **"Пустые строки"** определяет, допускается ли ввод в данное поле пустых строк (строк, не содержащих символов).

Свойство **"Пустые строки"** может иметь следующие значения (таблица 2.3):

Таблица 2.3.
Значения свойства «Пустые строки»

Значение	Описание
Да	Пустые строки являются допустимыми значениями.
Нет	Пустые строки не являются допустимыми значениями.

При задании значения "Да" для свойств **"Пустые строки"** и **"Обязательное поле"** Microsoft Access различает несуществующие данные (сохраняются в виде пустых строк) и данные, которые существуют, но не известны (сохраняются в виде пустых (Null) значений).

Для различия пустых строк от значений Null можно использовать свойство **"Формат поля"** (Format). При этом вместо пустых строк можно выводить строку "Отсутствуют данные".

В нашем примере значение поля "ФИО" должно присутствовать всегда и не может содержать пустые строки.

Сохранение описания таблицы

После того, как описание таблицы завершено, его надо сохранить. Этого можно достигнуть разными путями: выбрать меню **"Файл/Сохранить"** или **"Файл/Закрыть"** (после чего на вопрос "Сохранить изменения макета или структуры?" ответить "да") или щелкнуть по кнопке **"Вид"** инструментального меню и выбрать **"Режим таблицы"** и на сообщение "Сначала необходимо сохранить таблицу. Сделать это сейчас?" ответить: "Да" (этот способ надо использовать тогда, когда Вы хотите сразу после описания структуры таблицы вводить данные в эту таблицу). В появившемся после указанных действий окошке следует ввести имя созданной таблицы.

Создание таблиц для контрольного примера

Аналогичные действия повторяются при создании остальных таблиц БД.

Создадим таблицы "КАФЕДРА" со структурой, представленной на рис. 2.19, и "ДЕТИ" со структурой, представленной на рис. 2.20.

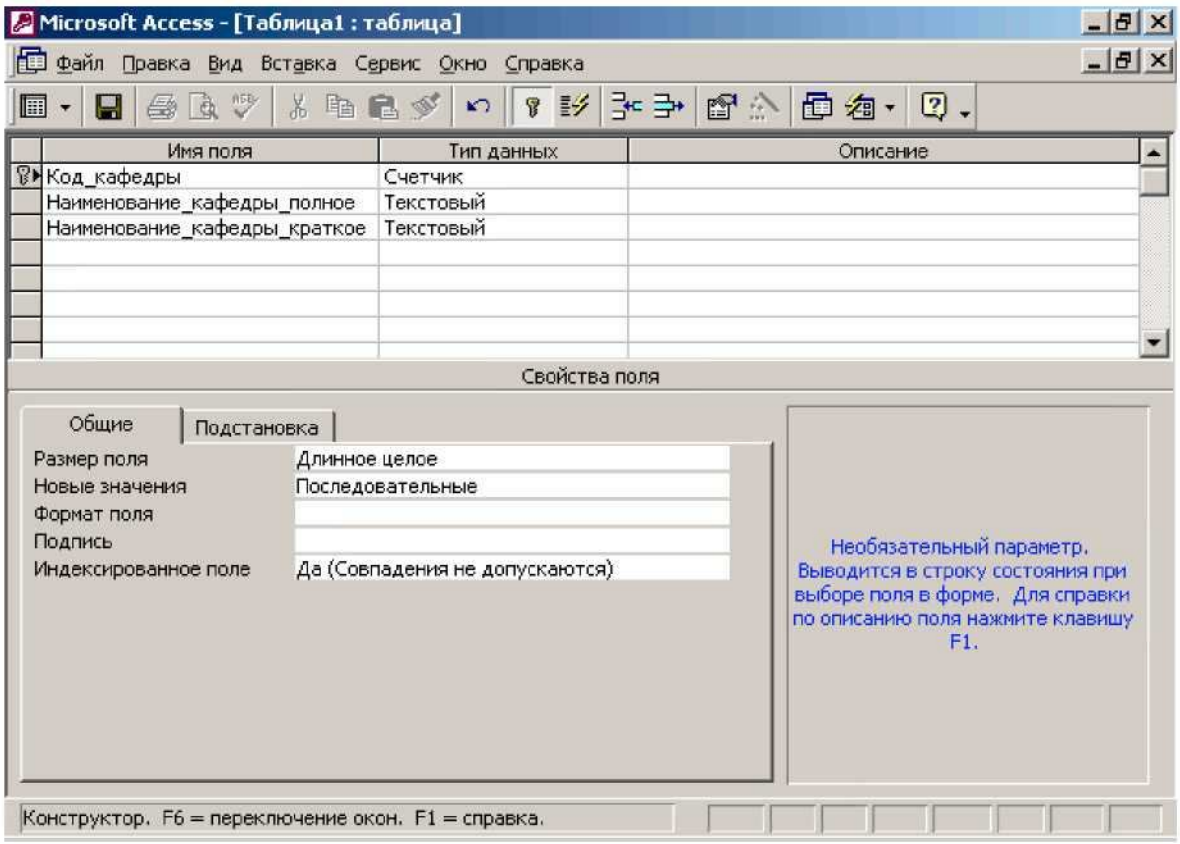


Рис. 2.19. Структура таблицы «КАФЕДРА»

При создании таблицы КАФЕДРА ключ «КОД_КАФЕДРЫ», как и в случае с таблицей «СОТРУДНИК», создадим автоматически при закрытии таблицы.

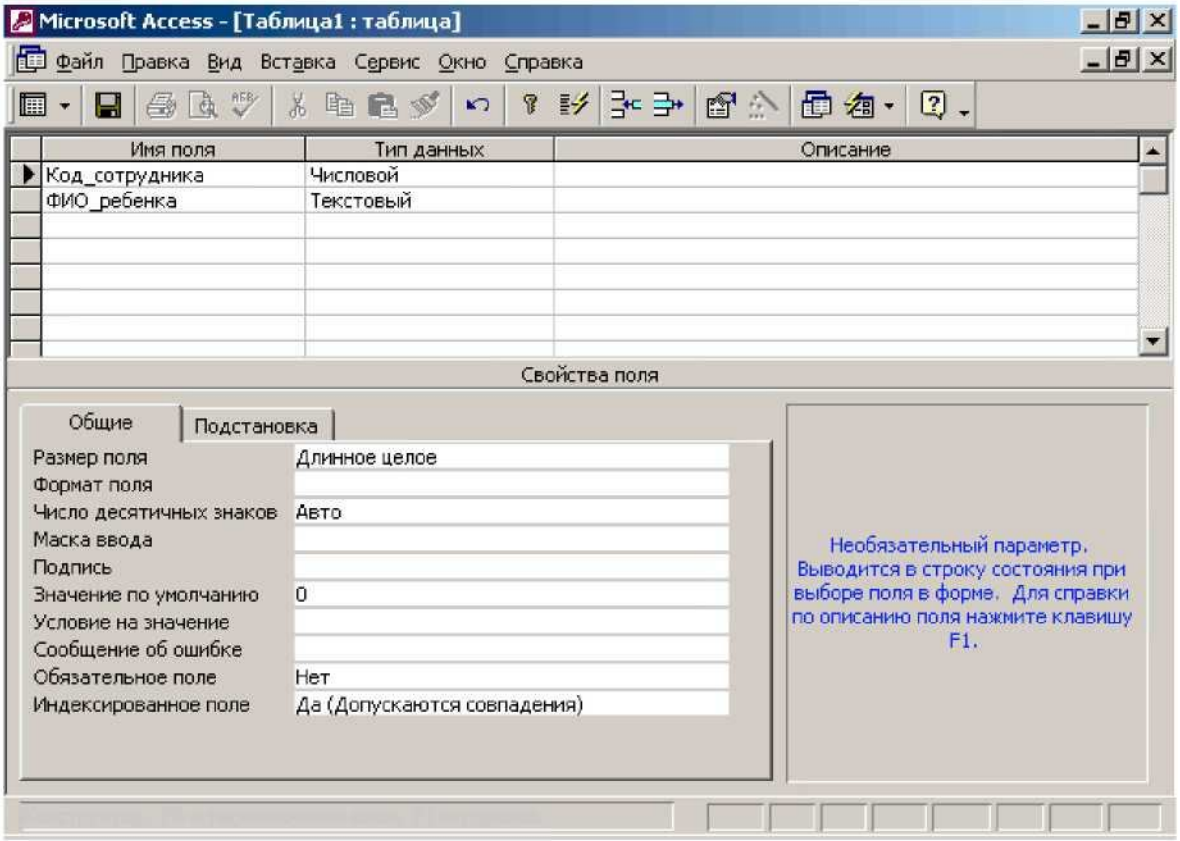


Рис. 2.20. Структура таблицы «ДЕТИ»

При создании таблиц следует помнить, что в реляционных базах данных связывание таблиц происходит по значениям соответствующих полей связи. Эти поля должны соответствовать друг другу по типу и длине. В нашем примере речь идет о полях «КОД_КАФЕДРЫ» в таблице "КАФЕДРА" и одноименном поле в таблице «СОТРУДНИК», и «КОД_СОТРУДНИКА» в таблицах «СОТРУДНИК» и "ДЕТИ". Если в основной таблице ключевое поле имеет тип счетчик, то в подчиненной таблице соответствующее поле связи должно иметь тип «числовое» и размер поля - «длинное целое».

Изменение структуры таблиц

Если вы ошиблись при описании структуры таблицы или по каким-либо другим причинам хотите изменить ее, то это можно легко сделать. Если вы уже вышли из процесса создания таблицы, но еще продолжаете работать с ней, то можно перейти обратно в режим **"Конструктора",** воспользовавшись кнопкой **"Вид".** Если нужная таблица закрыта, то ее можно открыть в режиме **"Конструктор"** и таким образом вернуться в окно описания таблицы.

Для добавления поля в таблицу выберите строку, над которой требуется добавить новое поле, и нажмите кнопку **"Добавить строки"** на панели инструментов, либо просто нажмите клавишу "Ins". Для добавления поля в конец таблицы выберите первую пустую строку и вводите в нее описание очередного поля.

Если таблица уже содержит данные, то до изменения типов данных и размеров полей рекомендуется сделать ее копию, так как несовместимость существующих данных с новым значением свойства **«Тип данных»** может привести к потере данных.

Другие способы создания таблиц

Если Вы создаете таблицу, структура которой имеет много общего со структурой ранее созданной таблицы, то можно скопировать структуру существующей таблицы (для этого надо позиционироваться на соответствующей таблице, выбрать позицию меню **"Правка/Копировать",** потом - **"Правка/Вставить",** после чего в появившемся окне (рис. 2.21) ввести имя вновь создаваемой таблицы, а в качестве параметра вставки выбрать **"только структура").** Структура созданной таким образом таблицы может быть впоследствии скорректирована обычным способом. В приведенном примере в базе данных учебного заведения на основе таблицы "СОТРУДНИК" строится таблица "АСПИРАНТ".

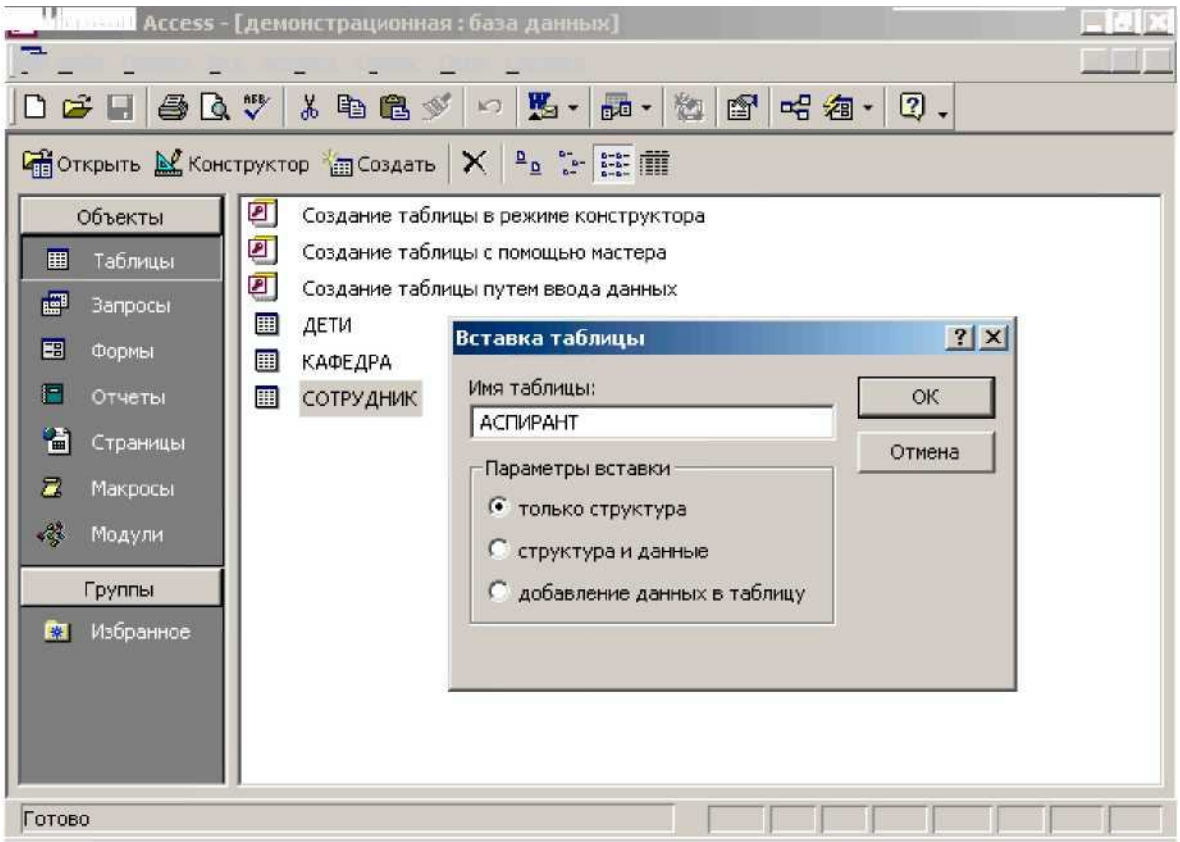


Рис. 2.21. Копирование структуры таблицы

Кроме того, создать таблицу можно с использованием "Мастера таблиц" **("Таблица/Создать/Мастер таблиц"). В** левой части окна **"Создание таблиц"** высвечивается перечень образцов таблиц, из которых вы можете выбрать подходящую по содержанию таблицу (рис. 2.22).

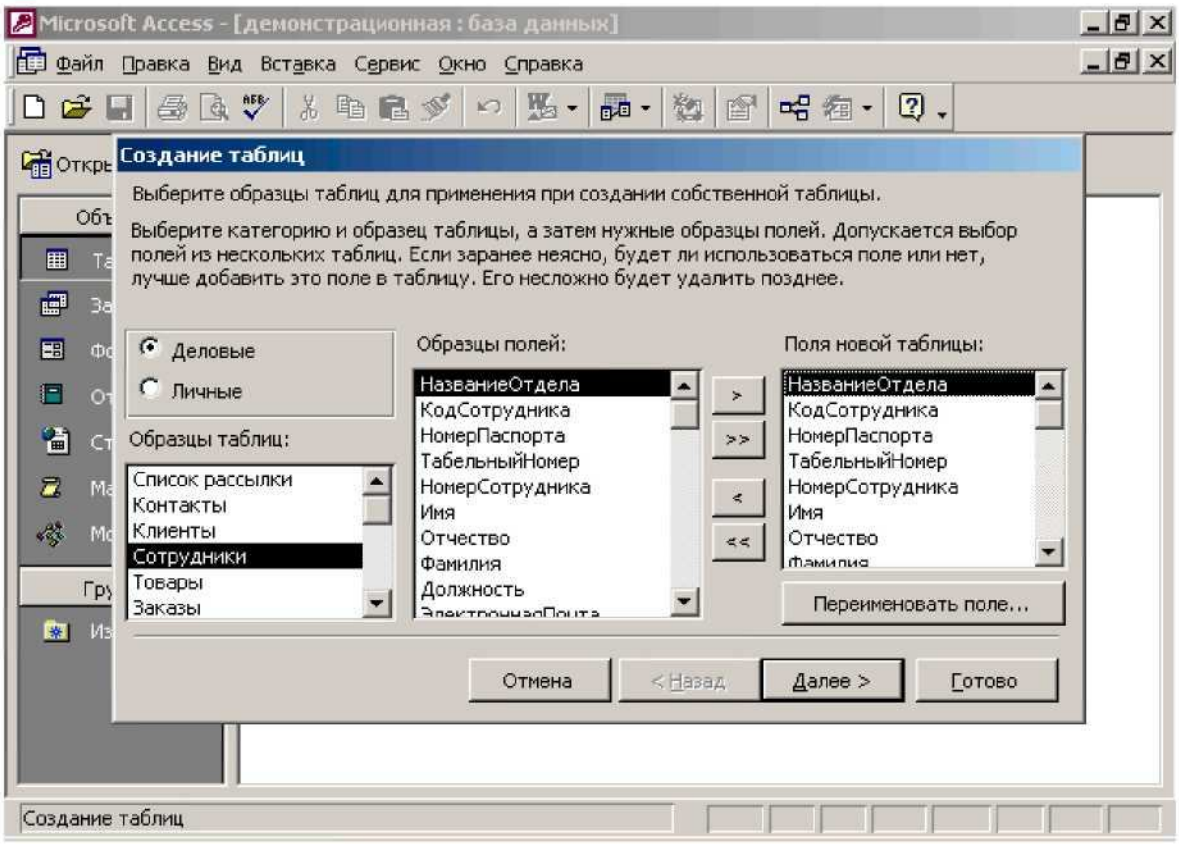


Рис. 2.22. Создание таблиц с использованием Мастера таблицы

Из выбранной таблицы Вы можете перенести все или некоторые поля во вновь создаваемую таблицу, можете изменить имя поля. Чтобы ввести какие-либо другие изменения в структуру создаваемой таблицы, следует завершить формирование таблицы с помощью мастера, после чего откорректировать структуру в обычном порядке. Как мы видим, использование этой возможности не освобождает от понимания основ проектирования БД, так как следует внимательно оценить, насколько предлагаемое в качестве образца решение соответствует вашим потребностям, и, при необходимости, изменить предлагаемую структуру БД.

Создать таблицу можно и путем импорта ее из других систем. Кроме того, в виде таблицы можно сохранить результат запроса. В Access имеется еще возможность создавать таблицу в режиме таблицы.

Связывание таблиц

После того, как таблицы созданы, можно задать их связанность. Для этого надо выбрать позицию меню **"Сервис/Схема данных"** (либо нажать соответствующую кнопку на панели инструментов). Далее, в открывшемся окне **"Схема данных"** следует добавить в окно те таблицы, между которыми будет

определяться связь. Таблицы, между которыми определяется связь, чаще всего, связаны отношением 1 :М. Для установления связи надо позиционироваться на поле связи (обычно это первичный ключ) в основной таблице (та, которая стоит на стороне "1") и, не отпуская клавишу мыши, перетащить появившийся значок на соответствующее поле в "зависимом" файле и отпустить клавишу мыши. После этого на экране появится окно **"Связи"** (рис. 2.23). Далее следует определить, надо ли задавать ограничения целостности связи, и если да, то выбрать режимы корректировок (обновления и удаления). Если вы задаете ограничения целостности, то поле связи основной записи должно быть проиндексировано.

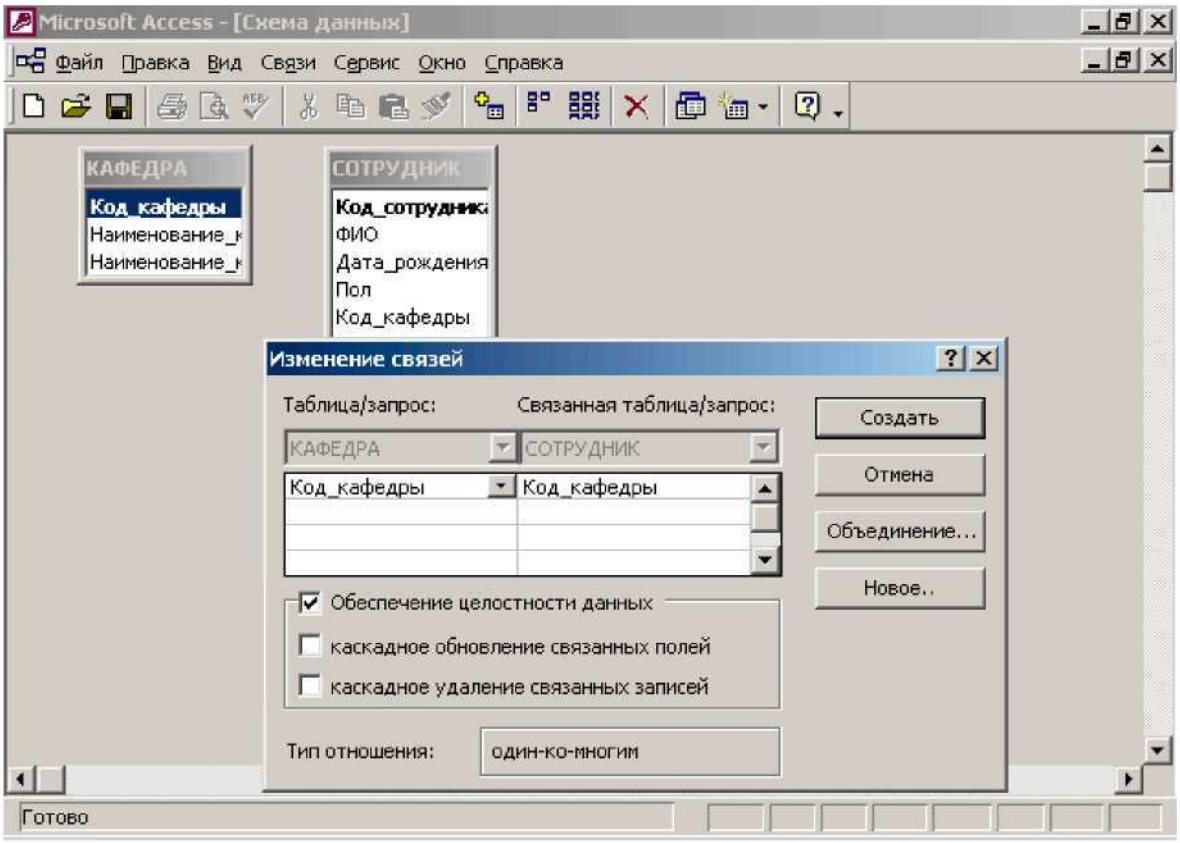


Рис. 2.23. Задание связи и ограничений целостности по связи

В рассматриваемом нами в качестве примера случае при связывании таблиц "КАФЕДРА" и "СОТРУДНИК" ограничение целостности следует задать, чтобы в таблице "СОТРУДНИК" не появлялись коды кафедр, которые отсутствуют в справочнике "КАФЕДРА". Задавать **"каскадное обновление связанных полей"** в данном случае не имеет смысла. Каскадное обновление означает, что при изменении первичного ключа в основной таблице, соответствующие поля в связанной таблице автоматически изменяются. В таблице "КАФЕДРА" поле "КОД_КАФЕДРЫ" имеет тип "Счетчик". Это означает, что изменять значение этого поля нельзя. Задавать каскадное удаление в данном случае тоже опасно, так как в случае ликвидации "КАФЕДРА" окажутся удаленными все записи сотрудников, работавших на этой кафедре.

Для связи же таблиц "СОТРУДНИК" и "ДЕТИ" (рис. 2.24) каскадное удаление вполне уместно.

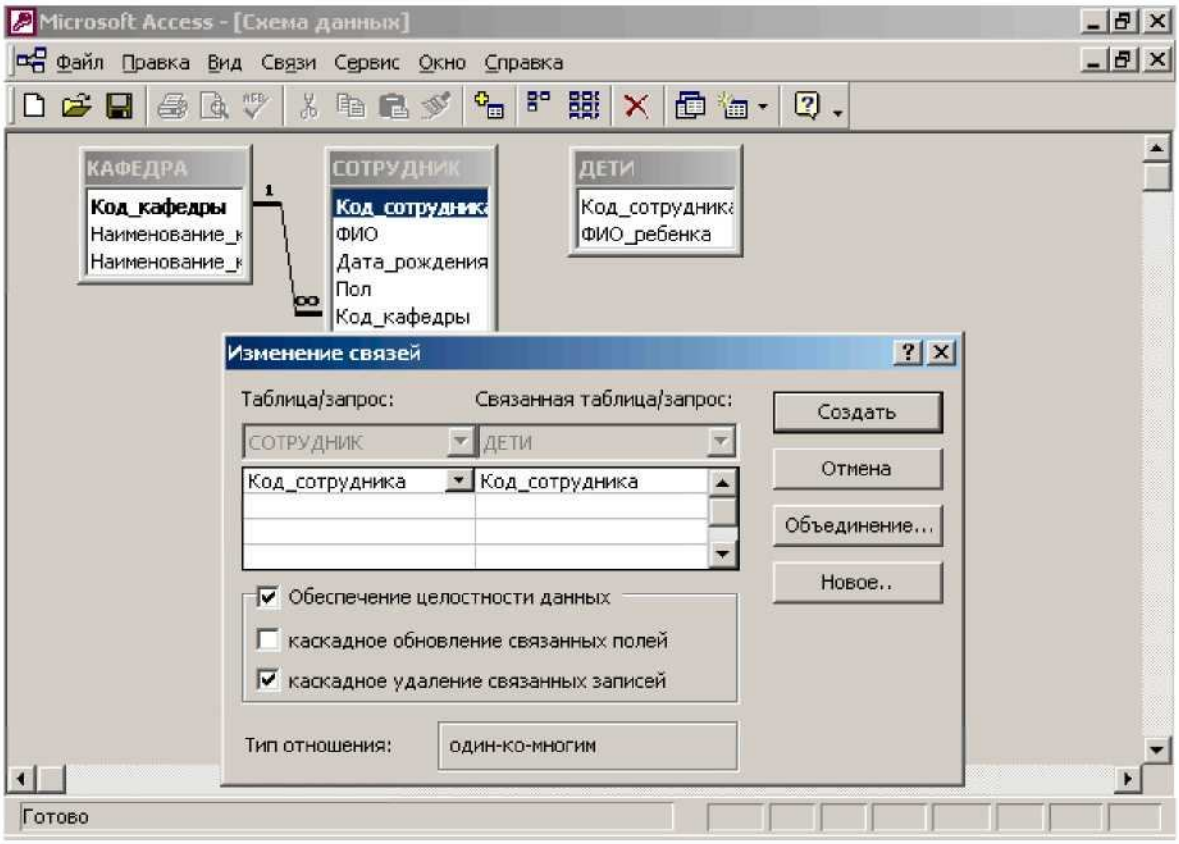


Рис. 2.24. Задание целостности связи и условий обновления связанных записей

Обеспечить ссылочную целостность можно и иным способом - используя поле подстановки: если значения будут переноситься из связанной таблицы, то в подчиненной не может появиться значение, отсутствующее в основной таблице.

Существуют понятия внутреннего, левого и правого соединения. В окне **"Связи",** появляющемся при установлении связи между двумя таблицами, есть кнопка **"Объединение",** нажав на которую пользователь попадает в окно **"Параметры объединения",** в котором он может выбрать одну из трех альтернатив:

1. объединение только тех записей, в которых значения связанных полей обеих таблиц совпадают;
2. объединение всех записей первой таблицы и только тех записей из второй таблицы, в которых значения связанных полей обеих таблиц совпадают,
3. объединение всех записей второй таблицы и только тех записей из первой таблицы, в которых значения связанных полей обеих таблиц совпадают.

Первая из перечисленных альтернатив обозначает ***внутреннее,*** вторая - ***левое,*** третья - ***правое*** соединение.

2.1.3. Задание ограничений целостности

Обеспечение целостности БД является одной из важнейших задач при создании БнД, так как обеспечение адекватности базы данных отображаемой предметной области является одним из основных требований, предъявляемых к БнД.

При изложении вопросов создания и связывания таблиц мы уже касались некоторых аспектов обеспечения целостности БД. Рассмотрим другие возможности задания ограничений целостности.

В Access многие ограничения целостности могут задаваться при создании таблицы.

Тип поля

Тип поля определяет допустимые символы, которые могут быть использованы при его заполнении. Для некоторых типов полей, например, поля типа «дата», осуществляется и более сложная проверка. Если допущена ошибка в типе данных или неправильно введена дата, то пользователь должен обязательно исправить ошибку, так как СУБД не дает других возможностей продолжить работу. Многие из свойств полей также позволяют обеспечивать контроль целостности. Такие свойства полей как:

размер поля
формат поля
маска ввода
значение по умолчанию
условия на значения
сообщение об ошибке
обязательное поле
пустые строки
индексированное поле, в той или иной степени связано с ограничениями целостности.

Поясним использование некоторых из перечисленных выше свойств в целях обеспечения контроля целостности на отдельных примерах.

Размер поля

В поле нельзя ввести больше символов, чем это зафиксировано в свойстве «размер поля» или предопределено типом поля.

Условия на значения

Одной из самых гибких возможностей определения ограничений целостности является задание "Условия на значения". Условия вводятся как выражения. Выражения могут быть простыми или сложными. Используя их можно задавать и диапазоны. Например, условие: >#1.92#, заданное как "Условие на значения" для поля "ДАТА_ПРИЕМА_НА_РАБОТУ", будет означать, что допустим ввод дат только после 1992 года. (Значения-даты необходимо заключать в символы номера (#)). Такое ограничение целостности может быть использовано, например, в случае, если организация, для которой ведется БД, была создана 1 января 2002 года, и все зачисления на работу были после этой даты. При задании такого ограничения целостности ввод значения в поле будет обязательным (даже если в свойстве поля «Условие на значение» зафиксировано - «нет»).

Условия на значения могут задаваться для полей или записей. Выражения, определяющие условия на значения, не должны содержать функции, определяемые пользователем, статистические функции или функции по подмножеству, функции CurrentUser или Eval, а также ссылки на формы, запросы и таблицы. Кроме того, выражение, указанное в качестве условия для поля, не должно содержать ссылки на другие поля. Выражение, указанное в качестве условия на значение для записи, может содержать ссылки на поля той же таблицы.

Условия на значения для записей задаются в окне свойств таблицы, вызываемом командой **"Свойства"** меню "Вид" в режиме конструктора таблицы.

Если пользователь задает значение свойства "Условие на значение", но не определяет свойство "Сообщение об ошибке", то при нарушении условия на значение Microsoft Access выводит стандартное сообщение об ошибке. Если значение свойства "Сообщение об ошибке" задано, то в сообщении об ошибке выводится текст, указанный в качестве значения этого свойства.

В Access нет специального способа задания домена перечислением. Как было показано выше, этого можно достичь, используя "Мастер подстановки". Кроме того, это можно сделать и путем задания соответствующего выражения для свойства ***Условия на значения.*** Например, для поля "Должность" в БД сотрудников вузов можно задать условие "ассистент" Or "старший преподаватель" Or "доцент" Or "профессор".

Microsoft Access автоматически накладывает условия на значение, определяемые **типом данных** поля, например, не допускается ввод текста в числовые поля.

Маска ввода

Предположим, Вы вводите в таблицу имена сотрудников. Для соответствующего поля можно задать маску ввода, которая позволит использовать только буквы при вводе, обеспечит преобразование первого символа в верхний регистр, всех остальных - в нижний, и допускающую использование не менее двух букв (считаем, что имен, состоящих из одной буквы, нет).

В Access такая маска ввода будет выглядеть следующим образом:

>L<L??????????????????????????

Символ "L" в маске обозначает, что в данную позицию **должна** быть введена буква, символ "?" обозначает, что в данную позицию **может** быть введена буква. Символ ">" преобразует все символы, расположенные правее этого знака, к верхнему регистру, символ "<"преобразует все символы, расположенные правее этого знака, к нижнему регистру.

Все символы, которые могут быть использованы в масках, и их назначение можно посмотреть в "Справочной системе" Access.

Использование подобных масок ввода не только обеспечивает контроль использования допустимых символов, но и облегчает процесс ввода данных.

Индексированное поле

Индексированное поле можно использовать для контроля на уникальность. В Access, как и во многих других системах, при определении для индексированного поля значения свойства "уникальный индекс" в это поле не допускается ввод повторяющихся значений.

В тех СУБД, которые поддерживают концепцию ключа (в том числе и в Microsoft Access), после того как в таблице определяется ключ, по этому полю производится индексирование и запрещается ввод повторяющихся или пустых значений ключа.

Как отмечалось выше, при задании связи между таблицами в Access можно установить флажок **"Поддержание целостности данных",** и в этом случае система будет автоматически поддерживать **ограничения целостности связи.**

При удалении основной записи, связанной с несколькими подчиненными, могут быть выбраны разные стратегии обновления:

- запретить удалять основную запись, если имеются подчиненные
- удалить вместе с основной записью и все подчиненные (каскадное удаление).

В Access, например, при поддержании целостности связи автоматически принимается первая стратегия. Чтобы отменить ее для данной связи надо установить флажок *"Каскадное удаление связанных записей".*

Для задания сложных условий можно использовать макросы или модули.

2.1.4. Ввод данных в базу данных

После того как завершено проектирование структуры базы данных, БД описана, можно приступать к вводу данных. Это можно сделать как сразу по окончании описания структуры таблицы, так и потом.

Ввод и корректировка данных в режиме "Таблица"

Как отмечалось выше, чтобы сразу после описания структуры таблицы вводить данные в эту таблицу надо щелкнуть по кнопке **"Вид"** и выбрать **"режим таблицы".** После сохранения описания таблицы, она высвечивается на экране в табличном виде (первая строка этой таблицы содержит имена полей таблицы, вторая - пустая, в которую и вводятся данные).

Для того чтобы попасть в режим **"Таблица"** для ввода данных в уже существующую таблицу надо в окне базы данных на вкладке **"Таблицы"** позиционироваться на строке, соответствующей названию требуемой таблицы, и нажать кнопку **"Открыть".** Каждая таблица содержит пустую запись, которая следует за последней существующей записью и предназначена для ввода новых данных (эта запись отмечена слева символом "звездочка" (*)). Позиционироваться на эту запись можно разными способами, например, нажав соответствующую кнопку в инструментальном меню или просто мышью. После чего следует ввести требуемые данные с клавиатуры.

В Access для рационализации процесса ввода данных в БД можно использовать свойство поля **«Значение по умолчанию».** Свойство **«Значение по умолчанию»** позволяет указать значение, которое будет автоматически вводится в поле при создании новой записи. В качестве значения по умолчанию чаще всего выбирается то значение, которое чаще всего встречается в записях БД. Например, для значения поля «Должность» в таблице, содержащей сведения о сотрудниках вуза, это будет «доцент».

Обычно в качестве значения по умолчанию указывается постоянное значение, однако, можно использовать и выражение. Например: для ввода текущей даты можно ввести выражение

=Date(), использующее функцию «Date()», выводящую текущую дату.

Если функция используется в выражении по умолчанию, то значение соответствующего поля может быть в последствии изменено вручную.

Выражения, которые используются в качестве значений по умолчанию, не должны содержать ссылки на элементы управления и другие поля, а также функции, определенные пользователем.

Выражения могут записываться непосредственно или строиться с помощью *"Построителя выражений"*.

Надо с осторожностью относиться к использованию значений по умолчанию.

Использование масок для ввода данных

Об использовании масок ввода уже немного говорилось в разделе «Создание таблиц». Рассмотрим некоторые другие примеры. Можно использовать маски для ввода конфиденциальной информации (если использовать маску типа "пароль", то вместо символов, введенных в поле, на экране будут изображаться звездочки (*)).

Если, например, в институте принято обозначение студенческих групп, включающее две заглавные буквы, дефис и три цифры, то для этого поля можно использовать следующую маску ввода:

>LL\-000

При этом не надо будет переключаться при вводе в верхний регистр, в качестве двух первых символов можно будет ввести только буквы, а последних трех - только цифры. Знак «-» вводиться и храниться в записях БД не будет, он присутствует только в маске при вводе и выводе данных.

Для ускорения ввода данных в текущее поле таблицы могут быть использованы определенные комбинации клавиш (таблица 1.5):

Таблица 2.4.

Назначение клавиш ускоренного ввода данных

Клавиша	Действие
Ctrl-;	Вводит текущую дату
Ctrl-:	Вводит текущее время
Ctrl-Alt-пробел	вводит значение поля установленное по умолчанию
Ctrl-' (апостроф) или ″ (кавычки)	вводит значение того же поля из предыдущей записи

Запись автоматически сохраняется при переходе к другой записи.

2.2. Запросы в СУБД Access

После описания таблиц и заполнения их данными к базе данных можно формулировать разнообразные запросы. Для задания запроса в Access следует перейти к закладке **«Запрос»** в окне базы данных. Для создания нового запроса следует нажать кнопку **«Создать»,** в результате чего появится окно **«Новый запрос»** (рис. 2.25).

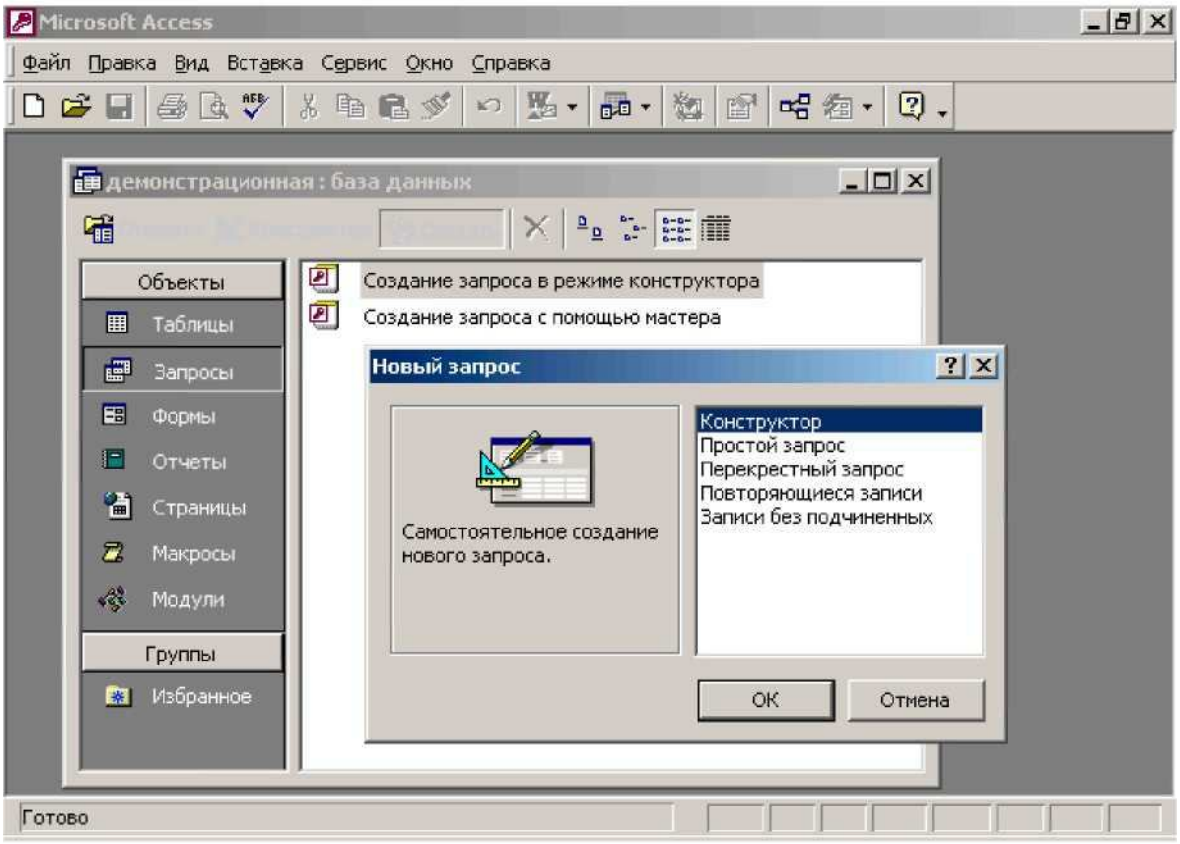


Рис. 2.25. Окно «Новый запрос»

Существует несколько способов создания запросов. Чаще всего используется возможность создания запроса с помощью **«Конструктора».**

Добавление таблиц в запрос

Первым шагом при создании запроса является определение таблиц, которые содержат исходную информацию. Допускается также создание запроса на основании других запросов или одновременно и таблиц, и запросов. Использование предварительно созданных запросов при создании нового запроса может помочь сделать сложный запрос, содержащий большое число взаимосвязанных таблиц и много разнообразных условий отбора, более простым для его формулирования. В некоторых случаях без разбиения запроса на несколько последовательно выполняемых шагов нельзя обойтись.

Если позиционироваться на строку **«Конструктор»** в окне нового запроса, то появиться окно **«Добавление таблицы»** (рис. 2.26), позволяющее выбрать таблицы/запросы, являющиеся источником данных для создаваемого запроса. Для того чтобы указать, на чем будет базироваться создаваемый запрос (таблице, запросе или том и другом одновременно), надо просто выбрать соответствующую закладку.

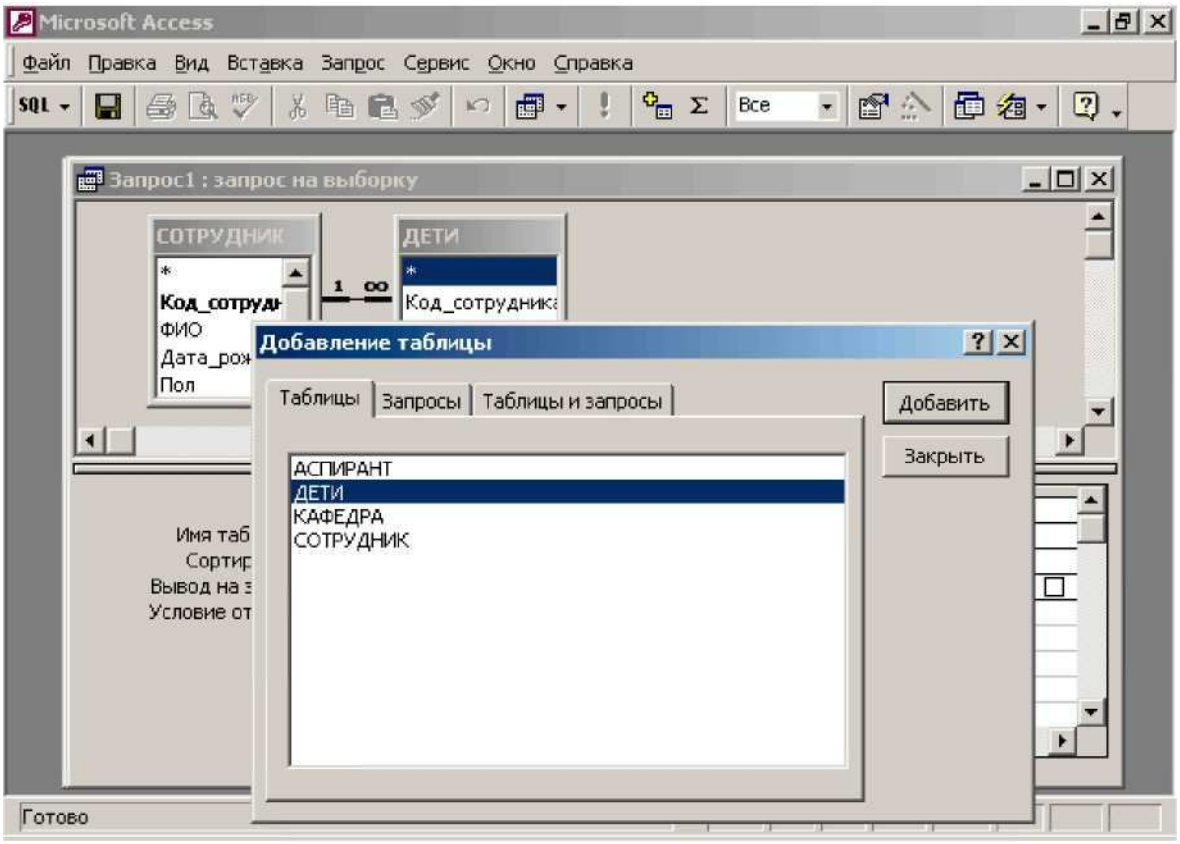


Рис. 2.26. Выбор источника запроса

Установив в появившемся списке доступных таблиц/запросов указатель на имя добавляемой таблицы или запроса, надо или выполнить двойной щелчок "мышью" или нажать клавишу **"Ввод".**

Допускается одновременное добавление в запрос нескольких таблиц или запросов. Для этого следует, удерживая нажатой клавишу Ctrl, выбрать имена добавляемых таблиц или запросов и нажать кнопку **"Добавить".**

В верхней части окна запроса выводится список полей добавленной таблицы или запроса (рис.2.26).

Возможно использование еще нескольких способов включения в запрос таблиц, на которых базируется запрос.

Для добавления таблицы можно в режиме конструктора запроса нажать кнопку **«Добавить таблицу»** на панели инструментов или выбрать в меню **«Запрос»** команду **«Добавить таблицу».** При этом открывается окно диалога **«Добавление таблицы».**

Пользователь имеет также возможность добавить в запрос таблицу или запрос, выбрав их имена в окне базы данных и переместив их с помощью мыши в верхнюю часть окна запроса.

Кроме того, можно позиционироваться на свободное место в верхней части окна запроса, нажать правую кнопку мыши и в появившемся ниспадающем меню выбрать позицию **«Добавить таблицу».**

Для добавления в запрос таблицы из другой базы данных или другого приложения следует сначала присоединить эту таблицу к активной базе данных.

Это присоединение выполняется путем использования команды **«Присоединить таблицу» (Меню Файл).**

Удаление таблицы из запроса

Если Вы ошибочно включили какую-то таблицу в запрос или по каким-либо иным причинам Вам надо удалить ранее включенную таблицу из запроса, то это легко можно сделать. Существует несколько способов удаления таблицы из запроса:

- Выбрать имя удаляемой таблицы или запроса в соответствующем списке и нажать клавишу **Del** или выбрать в меню **"Запрос"** команду **"Удалить таблицу".**
- Двойным щелчком мыши выделить нужную таблицу в соответствующей зоне экрана и затем нажать клавишу **Del.**

Имена полей удаленной таблицы или запроса удаляются из бланка запроса QBE. Удаление из запроса таблицы или запроса, на которых он базируется, не приводит к их удалению из базы данных.

Включение полей в запрос

После того, как Вы определили исходные таблицы/запросы, надо выбрать поля, используемые в создаваемом запросе. Существует несколько способов переноса поля в бланк запроса: "буксировка" с помощью мыши, двойной щелчок мышью на имени соответствующего поля в списке полей, выбор поля в раскрывающемся списке полей, который появляется, если нажать на знак «стрелки» в строке **«Поле»** бланка запроса.

Можно переносить в бланк запроса не по одному полю, а сразу требуемую совокупность полей. Выделение полей, подлежащих переносу, осуществляется стандартным для Windows-систем способом (нажать клавишу SHIFT для выделения полей, расположенных друг за другом, и,
удерживая ее, выбрать первое и последнее поле набора; выбирать поля при нажатой клавише CTRL для полей, располагающихся в произвольном порядке не подряд друг за другом).

Не все поля, которые используются при формировании запроса, должны обязательно отражаться в ответе. Так, например, поле может быть необходимо для задания условия отбора, но надобность в его появлении в ответе отсутствует.

Поля, выводимые в ответ, указываются в строке конструктора запроса ***"Вывод на экран".*** В соответствующих колонках этой строки указывается знак вхождения поля в ответ ("v" - "галочка").

Можно перенести в бланк запроса одновременно все поля. Для этого надо установить указатель на заголовок списка полей и дважды щелкнуть кнопкой мыши или установить указатель на символ звездочки (*) и нажать кнопку мыши (рис. 2.27).

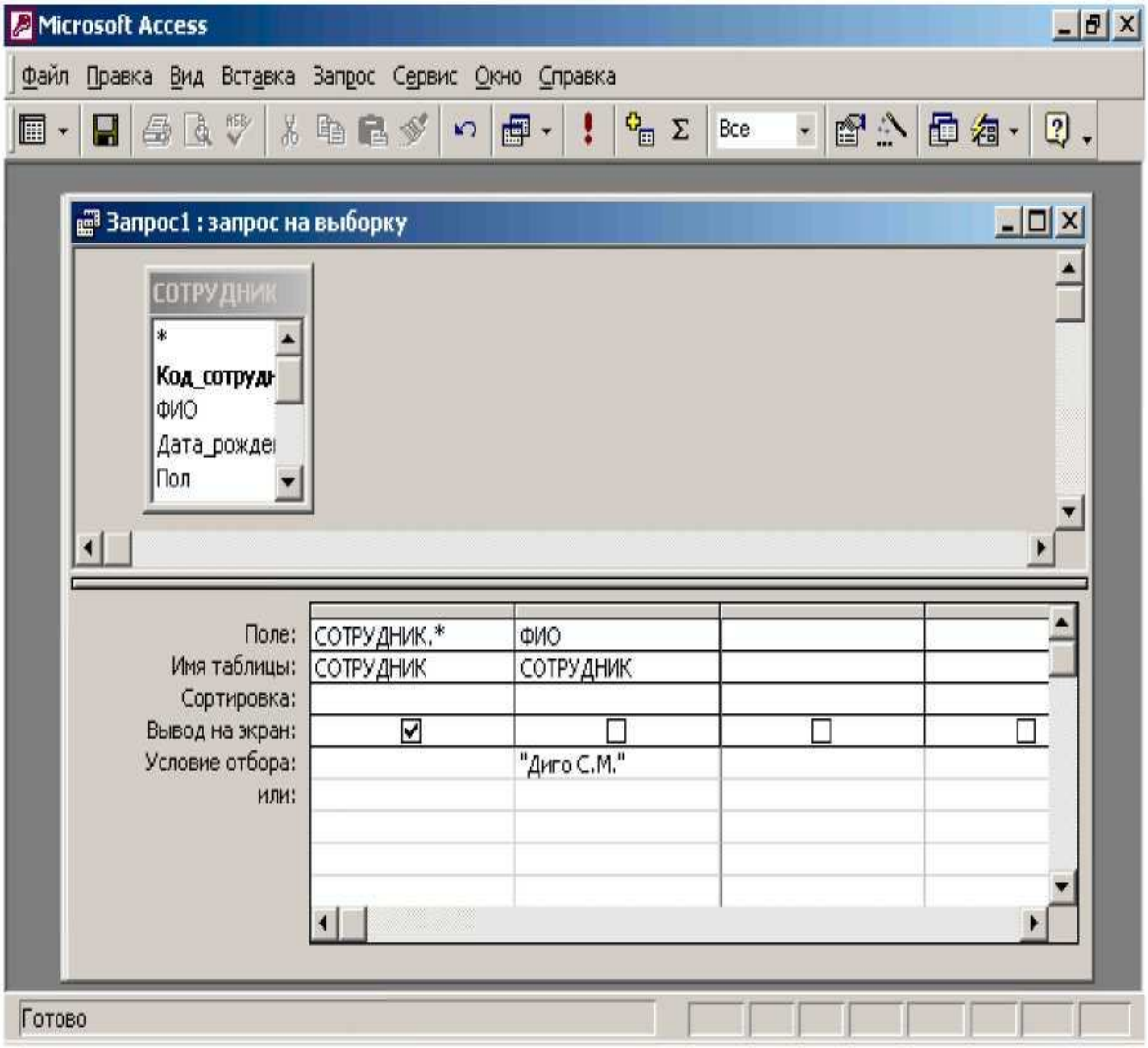


Рис. 2.27. Использование «*» в запросе

Есть разница, как поля были введены в запрос. При использовании символа звездочки в запрос автоматически включаются все поля, добавленные в базовую таблицу/запрос после создания данного запроса. Все удаленные поля будут автоматически удаляться из запроса. С одной стороны - это хорошо, с другой - может случиться, что пользователь в ответ на один и тот же запрос будет получать разный ответ, и, вполне может быть, не тот, который он ожидает. Так, например, если в таблице "СОТРУДНИК" первоначально фиксировались только основные данные по сотруднику, а затем было введено много других полей, то совсем не обязательно, что пользователь захочет видеть все эти данные, в ответ на свой запрос.

Если же поле, включенное в запрос явным способом, было впоследствии удалено из таблицы, то запрос может выполняться не совсем корректно.

Так как поля, включенные в запрос путем использования "*", в явном виде в бланке запроса не высвечиваются, то те поля, которые используются в условии отбора, надо дополнительно включить в бланк запроса. Чтобы эти поля дважды не выводились в ответ, надо у этих полей снять флажок ***«Вывод на экран»*** (рис. 2.27). Изображенный на рис. 2.27 запрос реализует вывод всех данных, содержащихся в таблице «СОТРУДНИК», по сотруднику с указанной фамилией, причем поле "ФИО" выводится в ответе только один раз, так как снят флажок "v" в колонке «ФИО». Изображенный на рис. 2.27 запрос реализует вывод всех данных, содержащихся в таблице «СОТРУДНИК», по сотруднику Диго С.М., причем поле "ФИО" выводится в ответе только один раз, так как снят флажок "v" в колонке «ФИО».

Обобщая выше сказанное, можно сделать вывод: поля включаются в бланк запроса в том случае, если они нужны в ответе, либо если они используются для задания условий отбора. В последнем случае они могут включаться, а могут и не включаться в ответ.

Задание условий отбора

Естественно, что при создании запросов важнейшим моментом является задание условий отбора. В предыдущем примере мы уже использовали условие отбора для получения информации по одному конкретному сотруднику. Язык QBE, реализованный в СУБД Access, относится к классу табличных двухмерных языков. Условие отбора необходимо задавать в таблице бланка запроса в той графе, к которой относится данное условие. На рис. 2.27 такое условие задано в графе ФИО.

Различают несколько типов запросов: *запрос на выборку* (Select), *перекрестный запрос* (Crosstab), *создание таблицы* (Make-table), *запрос на обновление* (Update), *добавление* (Append), *удаление* (Delete).

Управление выводом повторяющихся строк

В том случае, если в ответ выводятся не все поля исходной таблицы, может случиться, что строки в ответе будут повторяться. Например, если вывести только список кафедр из таблицы "СОТРУДНИК", то наименования одних и тех же кафедр могут встречаться несколько раз. Для того чтобы управлять выводом повторяющихся строк, можно позиционироваться на произвольное место вне бланка запроса и списка полей, нажать правую кнопку мыши, и в появившемся контекстном меню (рис. 2.28) выбрать строку "Свойства" (либо выбрать соответствующую кнопку на панели инструментов).

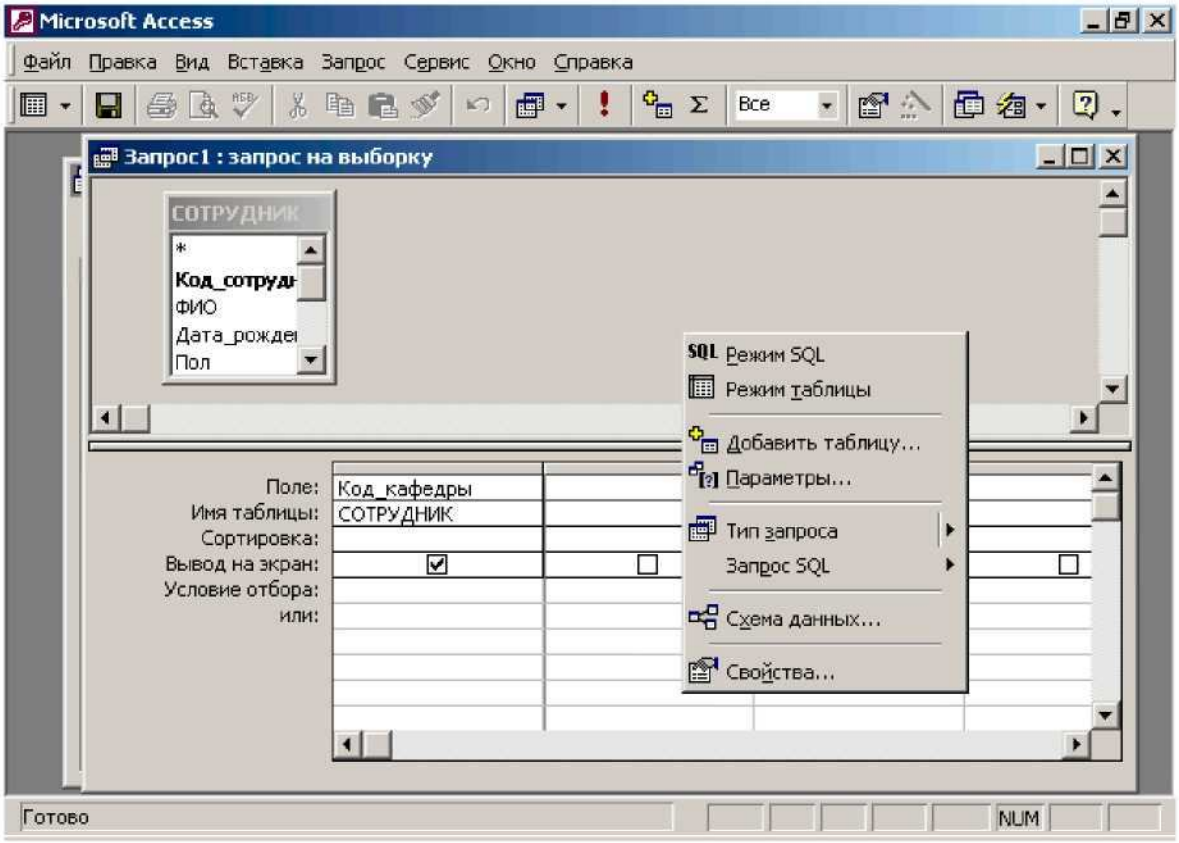


Рис.2.28. Контекстное меню поля в запросе

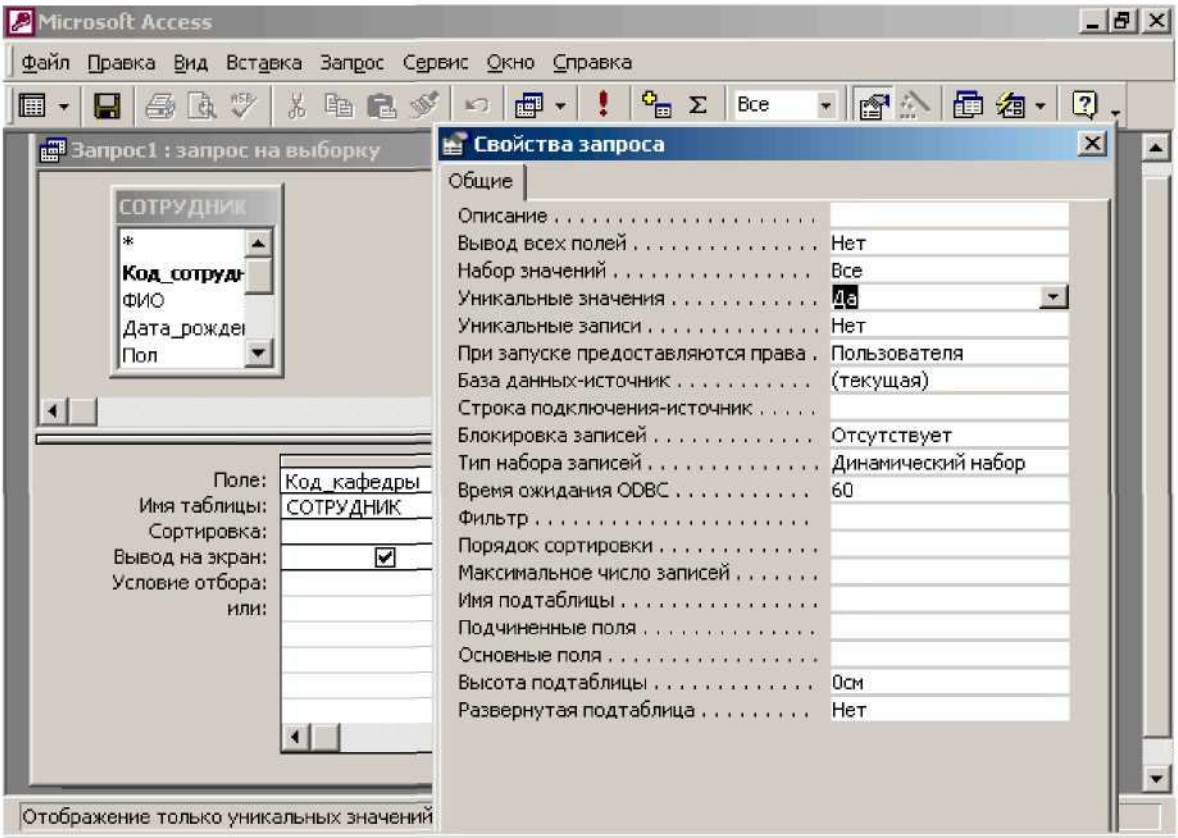


Рис. 2.29. Свойства запроса

Среди свойств запроса (рис. 2.29) есть два: **«Уникальные записи»** и **«Уникальные значения»,** которые служат указанным целям. Если вы хотите, чтобы

в ответ выдавался список кафедр без повторов, задайте для свойства **«Уникальные значения»** значение "Да".

Просмотр результатов выполнения запроса

Для того чтобы **посмотреть ответ** можно щелкнуть мышью на кнопке **«Запуск»** ("!") на панели инструментов, либо выбрать соответствующую возможность из меню **«Запрос/Запуск»,** либо щелкнуть на стрелку на кнопке "Вид" и выбрать из появившегося списка вид **«Режим таблицы».** Для того чтобы опять вернуться к построению/корректировке запроса, надо выбрать режим **«Конструктор»** (рис. 2.30).

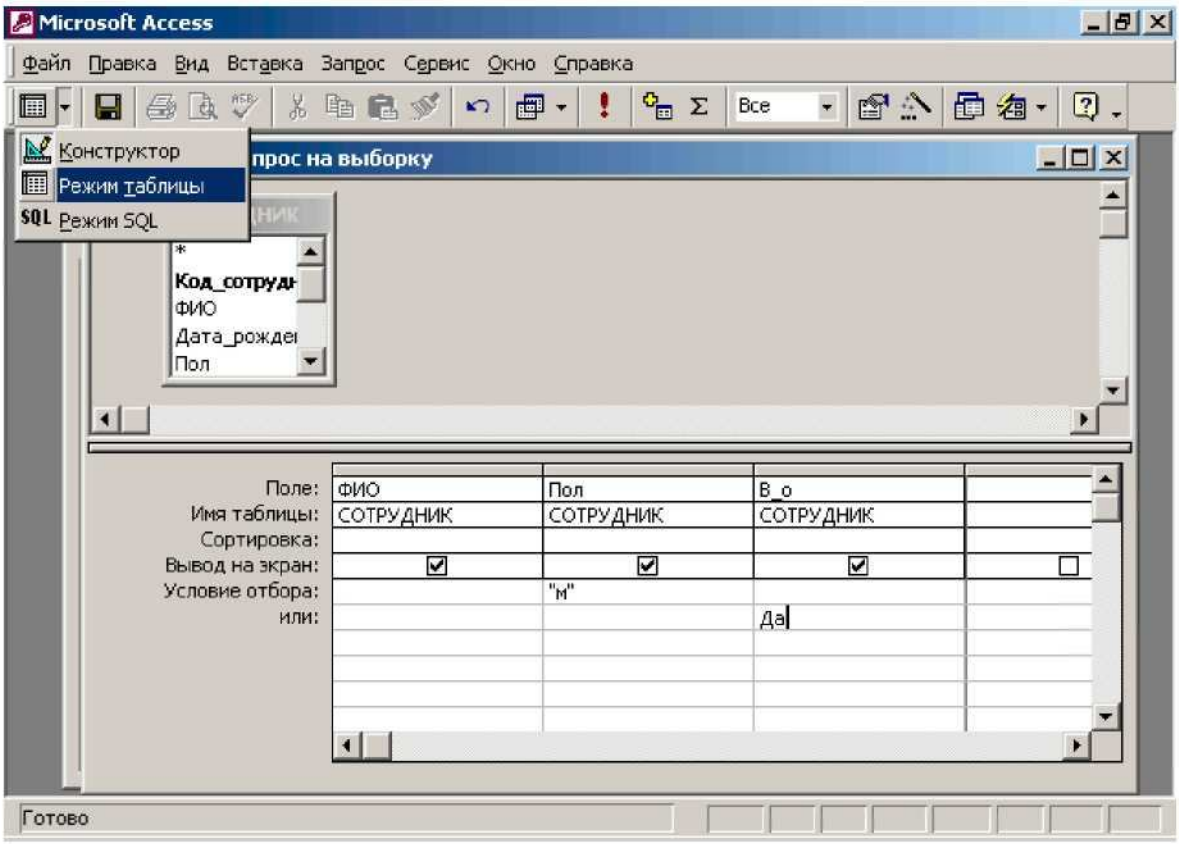


Рис. 2.30. Переключение режимов

Сохранение описания запроса

Любой запрос можно сохранить для последующего использования. Это можно сделать несколькими способами, например, выбирая позиции меню **«Файл/Закрыть»,** ответив «ДА» на вопрос о сохранении файла и задав после этого имя запроса. Сохраненный запрос можно впоследствии «открывать», что означает его выполнение. Сохраненный запрос может быть скорректирован, если открыть его в режиме конструктора.

Виды запросов

Наиболее часто используемым типом запросов является запрос на выборку. Именно с них мы и начнем изучение возможностей задания запросов в Access.

Простые запросы

Запросы с простыми условиями, включающими только один аргумент поиска, будем коротко называть *простым запросом*. При создании простого запроса условие отбора записывается в соответствующий столбец бланка запроса. Например, если надо отобрать информацию о конкретном сотруднике, то в столбец "ФИО" в строке **"условие отбора"** надо записать ФИО данного сотрудника (рис. 2.27).

Как известно, в большинстве СУБД, при вводе в выражение значений того или иного типа используются соответствующие данному типу данных ограничители. В Access при задании запроса ограничители можно не ставить. В зависимости от типа поля, которое вводится в выражение, определяющее условие отбора, ограничители добавляются системой автоматически:

прямые кавычки (" ") вокруг строковых значений.

символы (#) вокруг дат.

В столбце можно записывать не только значение атрибута, но и знак операции сравнения; по умолчанию принимается знак " = ". Если Вам, например, надо определить список всех сотрудников, имеющих оклад меньше 1000 руб., то запрос будет выглядеть так, как изображено на рис. 2.31.

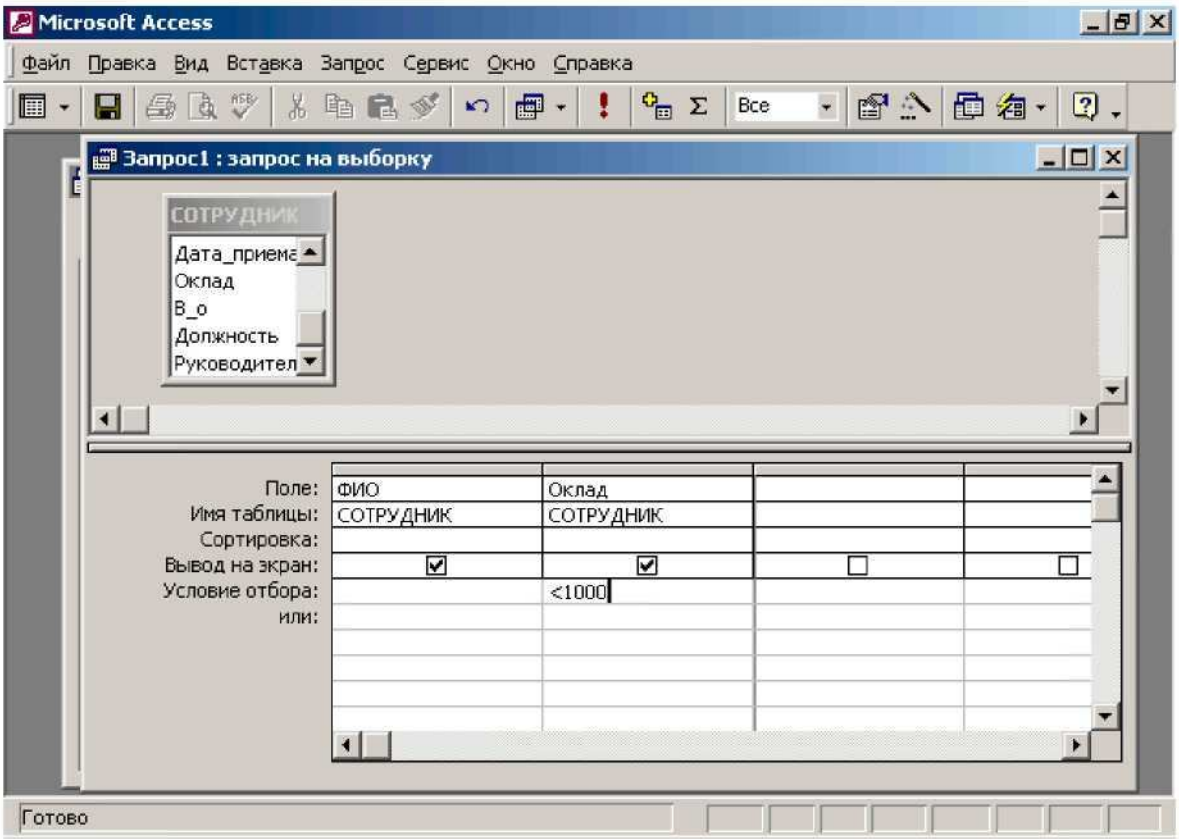


Рис. 2.31. Использование операторов сравнения при задании запроса

В условиях отбора можно задавать и диапазон значений. В этом случае запрос будет выглядеть подобно тому, как изображено на рис. 2.32.

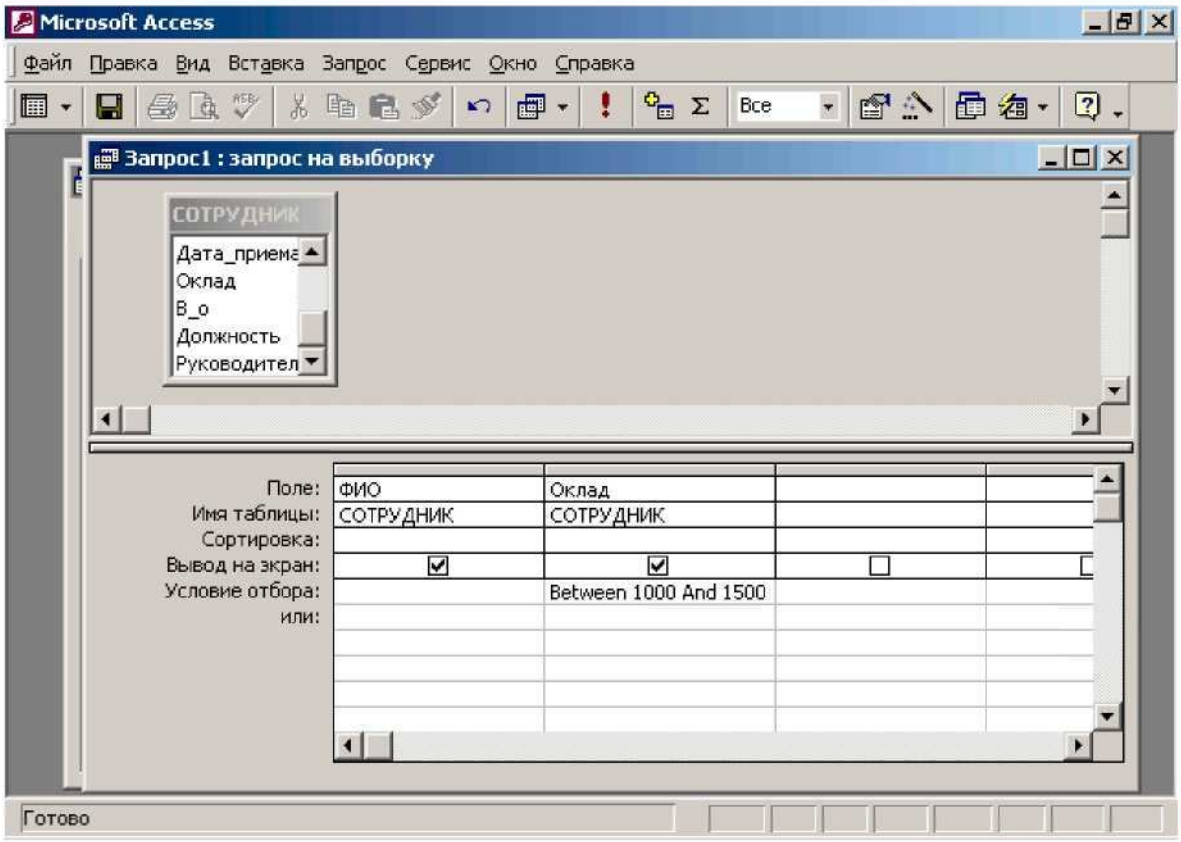


Рис. 2.32. Задание диапазона

Это же условие отбора в графе «ОКЛАД» можно было задать и следующим образом:

>=1000And<=1500.

В Access можно задавать и запросы с открытыми двусторонними диапазонами. Например, для выдачи списка сотрудников, получающих оклад меньше 1000 руб. и больше 15 000 руб. (т. е. мало- и высокооплачиваемых), условие отбора надо задать следующим образом:

<1000Or>15000.

Рассмотрим сложные запросы.

Если в условиях отбора используется несколько полей, то они могут соединяться оператором "И" либо "ИЛИ". Если аргументы поиска записаны в одной строке, то считается, что они соединены оператором "И" ("AND"). Если аргументы поиска записаны в разных строках, то считается, что они соединены оператором "ИЛИ" ("OR").

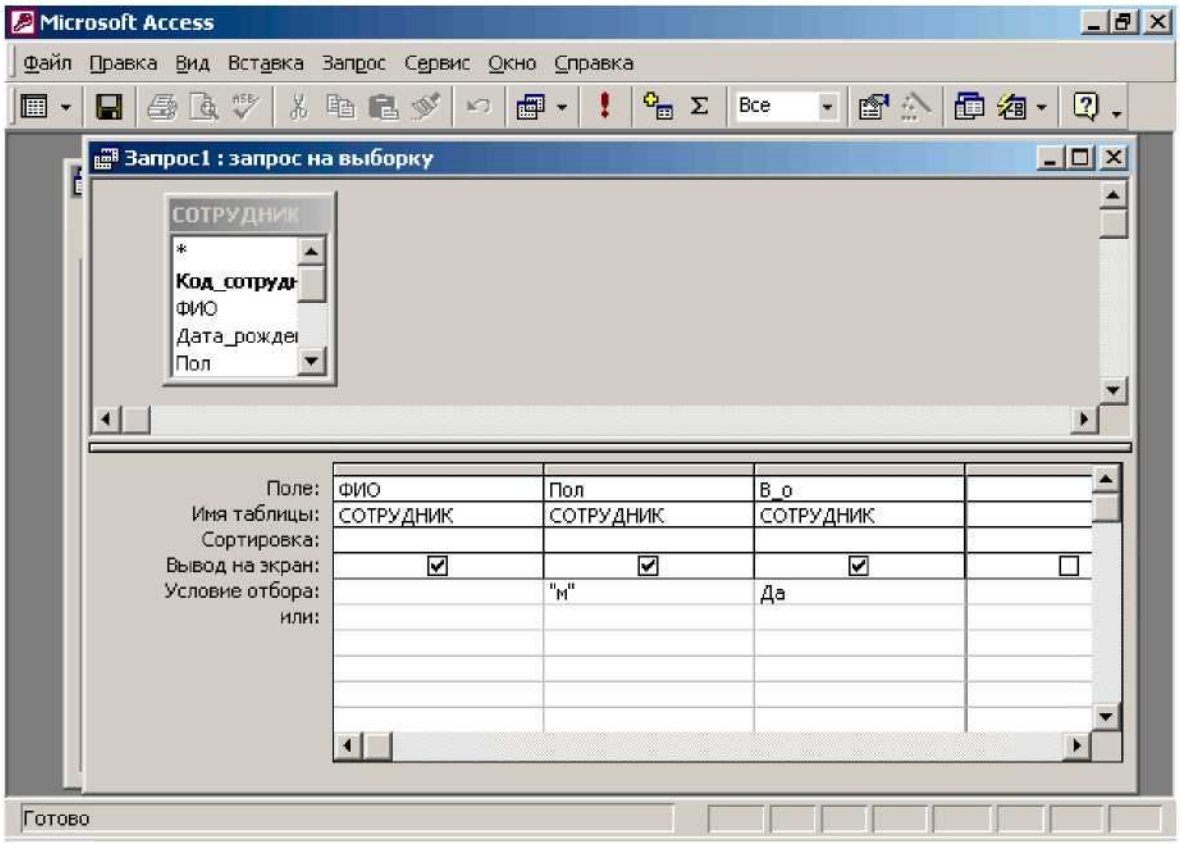


Рис. 2.33. Сложный запрос (оператор AND)

На рис. 2.33, 2.34 изображены примеры таких запросов. Первый из них выдает список военнообязанных мужчин (запрос "И"; аргументы запроса расположены на одной строке), второй (запрос "ИЛИ"; аргументы запроса расположены на разных строках) - всех мужчин и военнообязанных женщин.

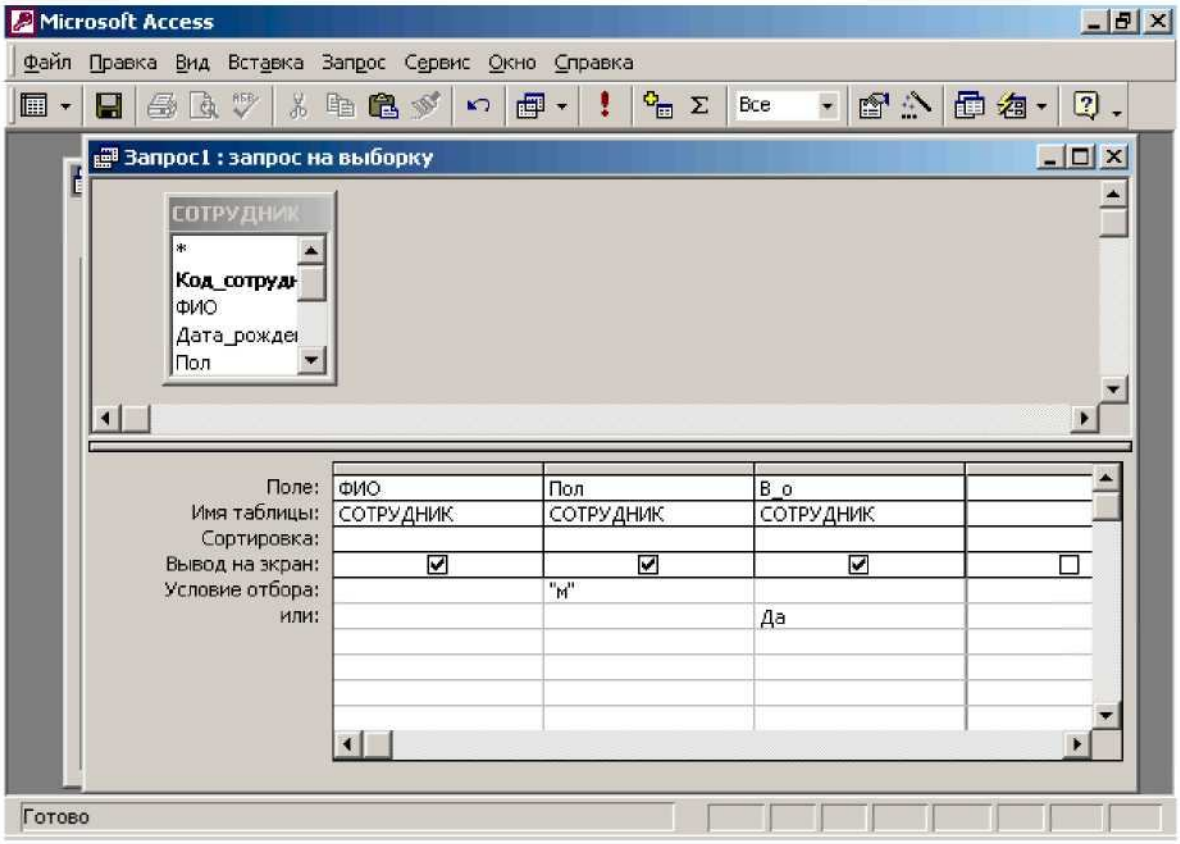


Рис. 2.34. Сложный запрос (оператор OR)

Рассмотрим запросы к связанным таблицам.

Если была предварительно определена схема данных, то при добавлении таблиц в запрос они будут должным образом связаны. Даже если связи между таблицами не были созданы пользователем предварительно, то при добавлении в запрос двух таблиц, содержащих поля с одинаковым или совместимым типом данных, а также, если одно из полей связи является ключевым, связи могут быть созданы автоматически. Автоматическое объединение (соединение) можно разрешить или запретить. Для этого надо выполнить следующую последовательность шагов:

В меню "Сервис" выбрать команду "Параметры"

Перейти к вкладке ***"Таблицы/Запросы".***

3. Установить/снять флажок ***"Автоматическое объединение".***

Параметр "Автоматическое объединение" относится только к новым запросам.

Если связи не были определены предварительно, и связи не созданы автоматически, то надо задать соединение таблиц вручную (так же, как это делалось при задании схемы).

Следует обратить внимание на то, что если связь не задана (и не отменено "Автоматическое объединение"), то будет осуществляться связь каждой записи одной таблицы с каждой записью второй таблицы.

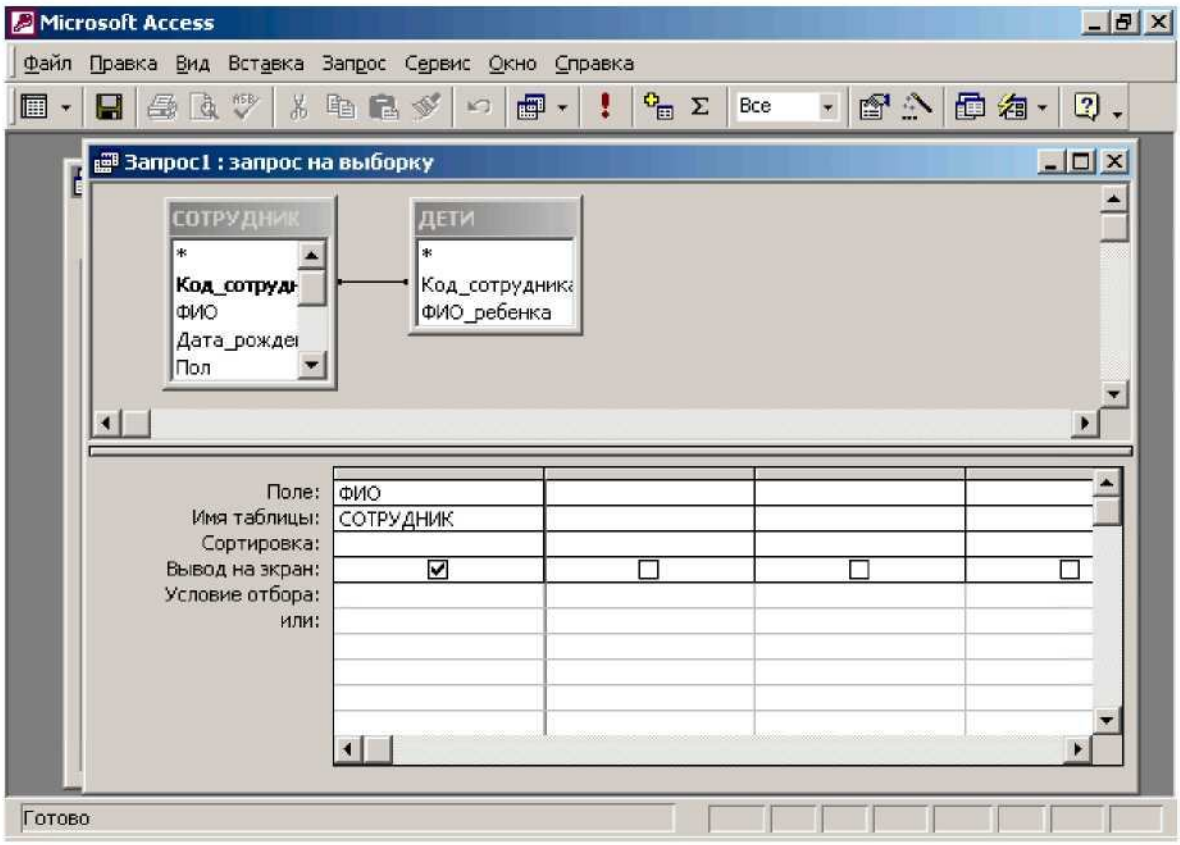


Рис. 2.35. Выполнение запросов на связанных таблицах

Надо осторожно относиться к формированию запросов к связанным таблицам. Что будет получено в ответ на запрос, изображенный на рис. 2.35? На самом деле ответить на этот вопрос, не имея дополнительной информации, нельзя. Чтобы ответить на поставленный вопрос надо знать, каковы параметры объединения (если Вы внимательны, то по виду линии сможете определить вид связи; сравните рис. 2.35 и 2.36) и какие значения имеют свойства **«Уникальные записи»** и **«Уникальные значения»** (этого на схеме не видно). Если задано обычное («внутреннее») соединение таблиц и для свойства «Уникальные значения» задано значение «Да», то в ответ на запрос, содержащий в бланке запроса поле «ФИО» и больше ничего, будет получен список сотрудников, имеющих детей.

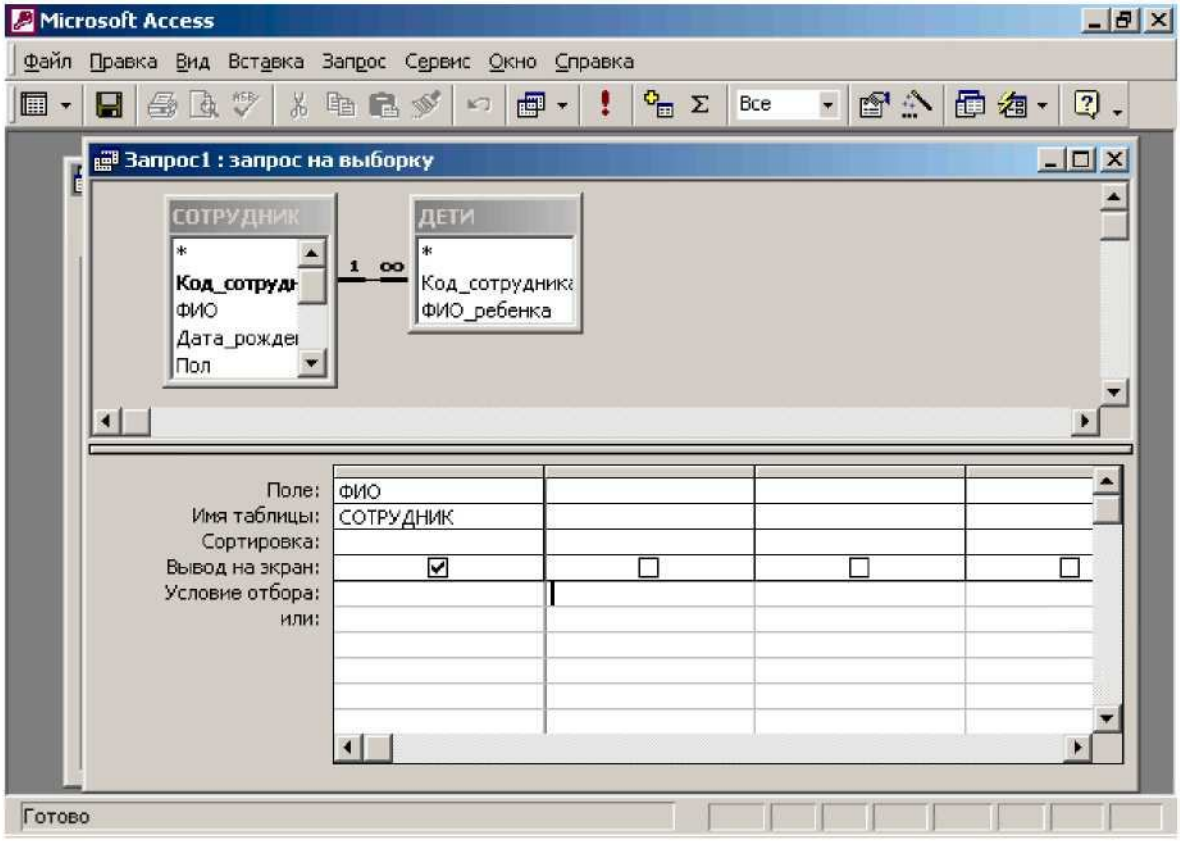


Рис. 2.36. Выполнение запросов на связанных таблицах (вариант 2)

При задании запроса желательно удалять из него все таблицы, поля которых не участвуют в формировании запроса.

При проектировании структуры базы данных тщательно продумывайте имена, которые вы даете полям разных таблиц.

Желательно также проверять связи, которые система задает автоматически.

Существуют понятия внутреннего, левого и правого соединения. В QBE Access это задается не в бланке запроса, а при задании схемы или при определении параметров связи в окне запроса. При формулировании запроса надо уточнить, какой тип объединения[4] был задан, и, при необходимости, изменить тип соединения, на тот, который необходим именно для этого запроса, так как тип объединения будет влиять на правильность ответа. Так, например, если необходимо выдать список всех сотрудников, а для тех, кто имеет детей - информацию о детях, то для соединения таблиц "СОТРУДНИК" и "ДЕТИ" надо выбрать вторую альтернативу в окне **"Параметры объединения".**

В реляционной теории различают операции "соединения" и "объединения". То, о чем идет сейчас речь, является реляционной операцией соединения. Но в системе Access обе эти операции называются "объединением", и когда рассматриваем, как это сделать в Access, то приходится переходить на терминологию этой системы.

Изменить тип объединения в запросе можно, выделив нужную связь и нажав на правую кнопку мыши. В появившемся контекстном меню (рис. 2.37) выбрать **«Параметры объединения».** Либо выбрать позицию меню **«Вид/Параметры объединения».**

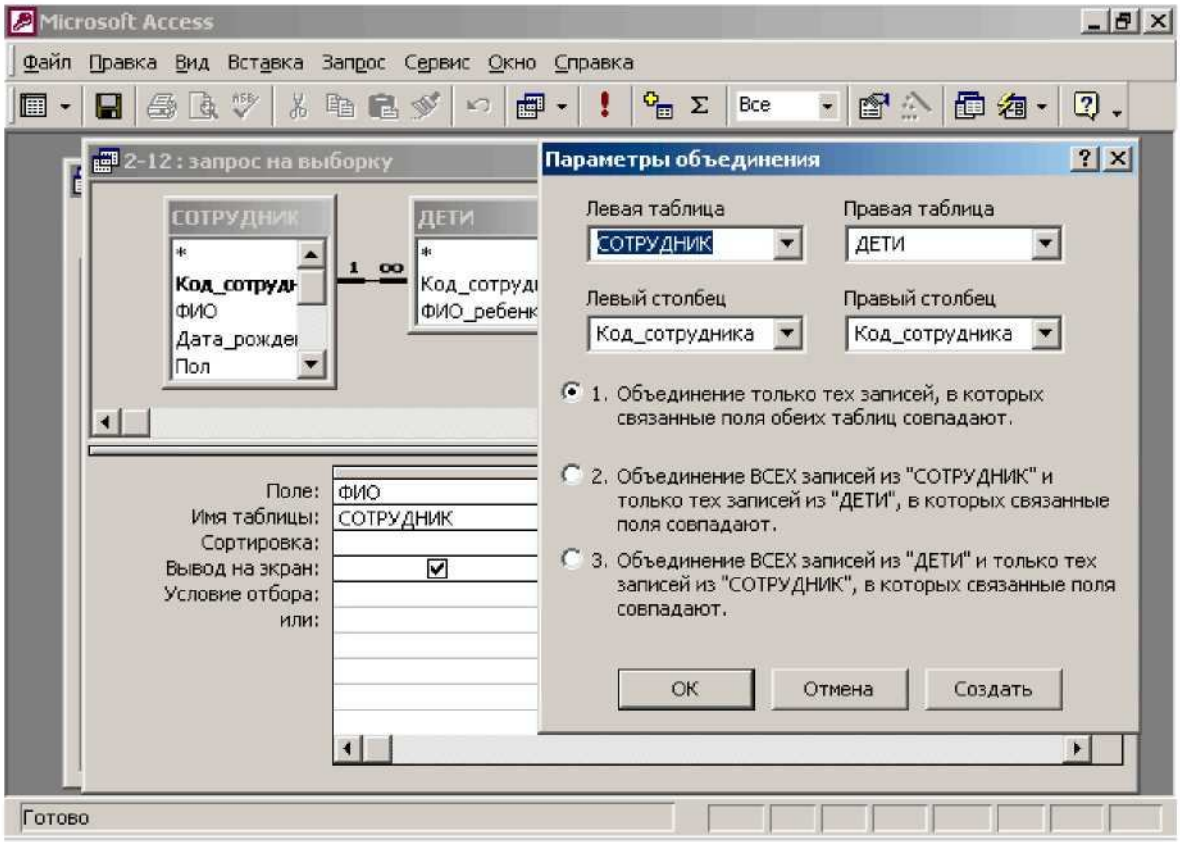


Рис. 2.37. Выбор типа объединения таблиц

Возможно создание запросов, в котором таблица соединяется сама с собой (так называемое «Самообъединение»). Например, для класса объектов «СОТРУДНИК» имеется связь «Быть руководителем». В рассматриваемом нами примере для отражения этой связи в таблицу «СОТРУДНИК» введено поле «Руководитель», которое содержит код сотрудника, являющего руководителем данного сотрудника.

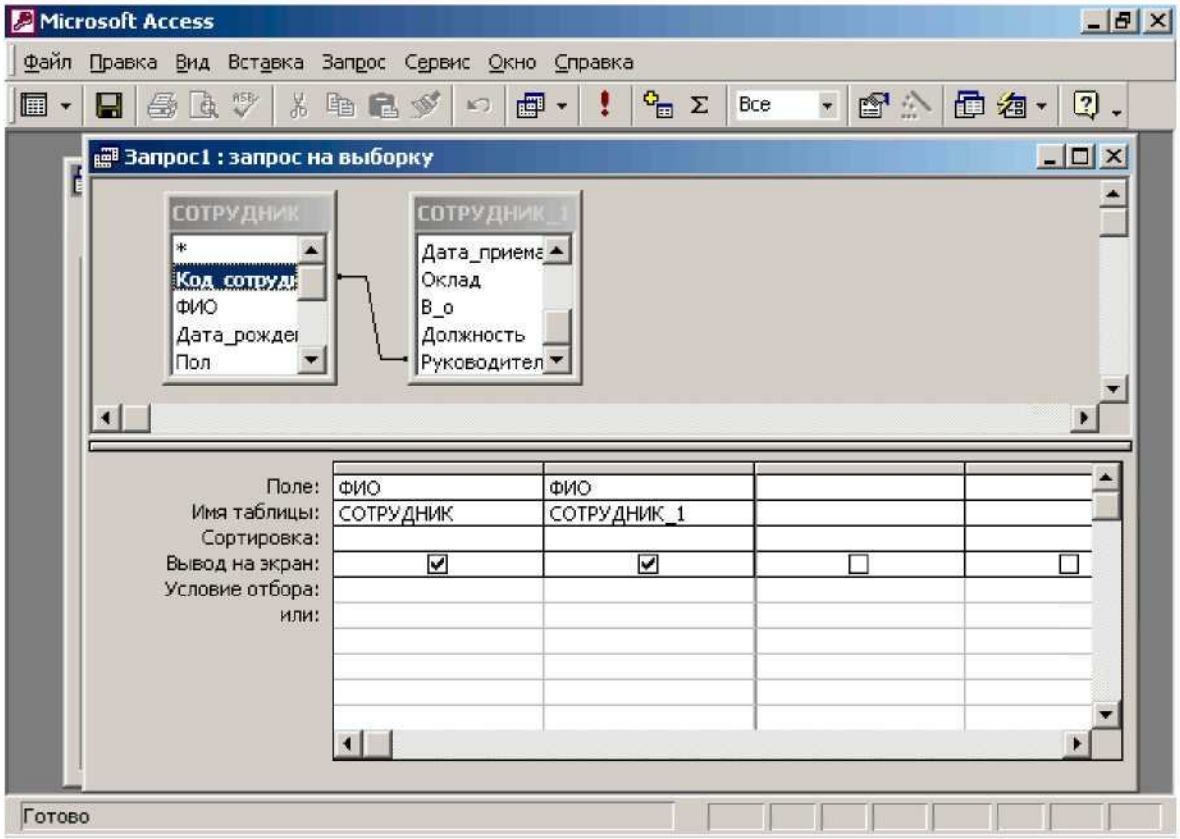


Рис. 2.38. Пример использования самообъединения таблиц в запросе

Для того чтобы объединить две копии одной и той же таблицы в запросе надо в режиме конструктора запроса дважды добавить эту таблицу в запрос. Далее надо осуществить соединение таблицы с ее копией обычным путем (установить курсор на поле связи в пером экземпляре таблицы и, не отпуская кнопки мыши, переместить появившийся значок на соответствующее поле в списке полей другой таблицы). На рис. 2.38 изображен запрос «Для каждого из руководителей выдать список его подчиненных». Для того чтобы в результатной таблице было понятно, что означает поле «ФИО» в каждом столбце, можно переименовать эти столбцы, назвав первый «Руководитель», второй - «Подчиненный». Для этого можно щелкнуть правой клавишей мыши на соответствующем поле, в высветившемся меню выбрать позицию **«Свойства»** и в появившемся окне **«Свойства поля»** в строке **"Подпись"** ввести требуемый заголовок столбца (рис. 2.39).

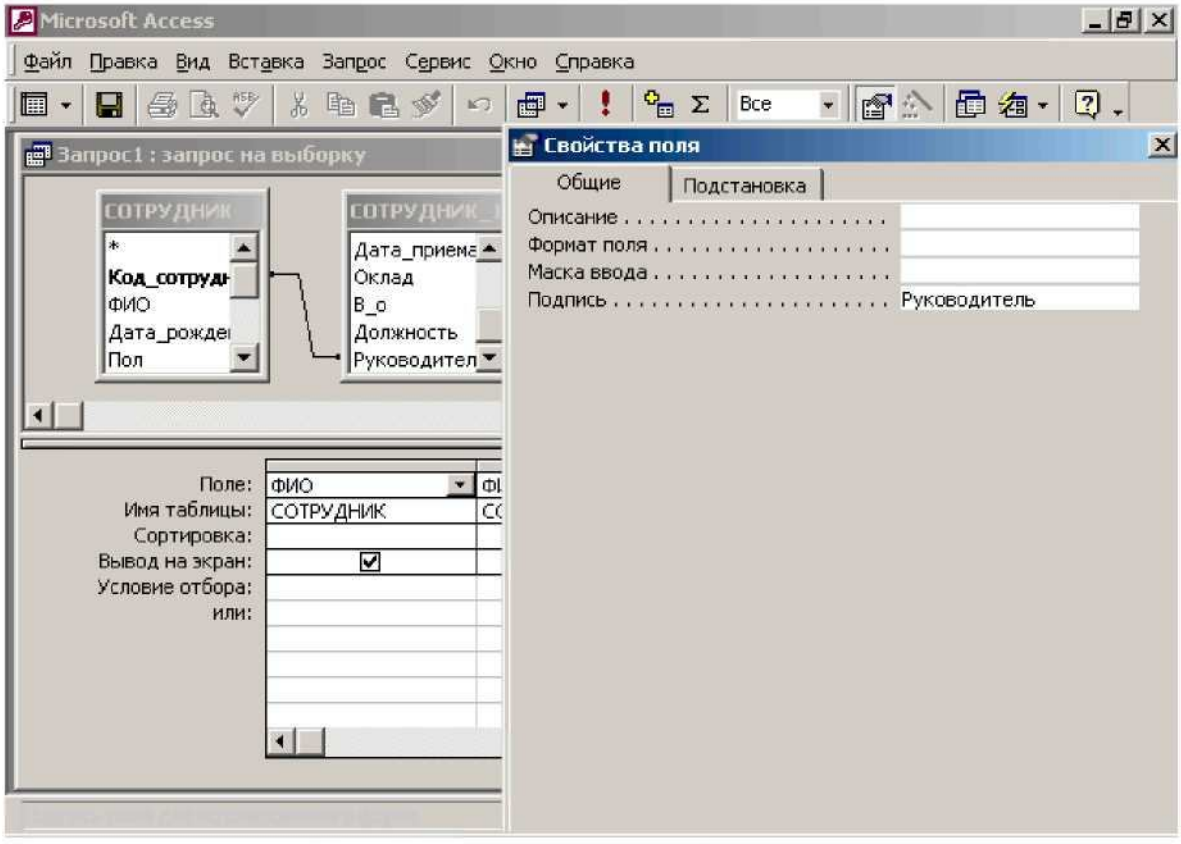


Рис. 2.39. Изменение подписи поля

Вид результатной таблицы после произведенных действий представлен на (рис. 2.40).

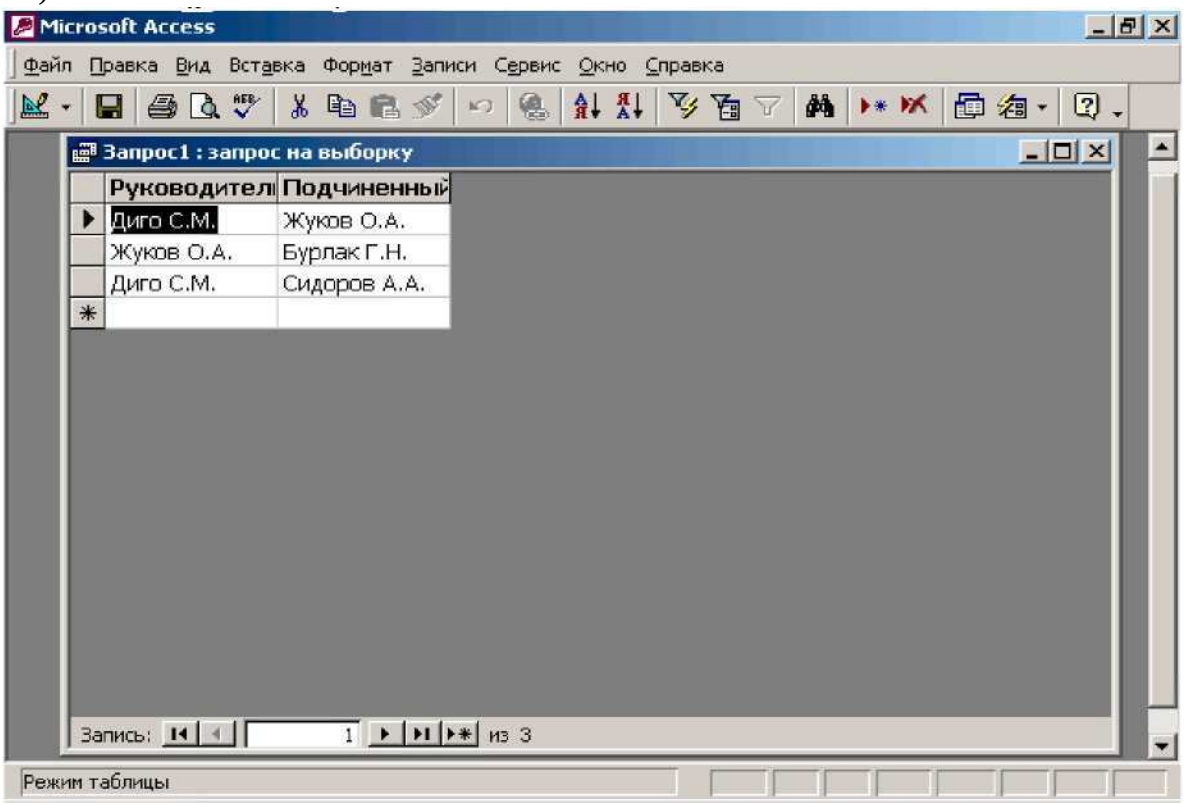


Рис. 2.40. Вид результатной таблицы (результат запроса с самообъединением таблицы)

Рассмотрим запросы с подгруппировкой.

Термина «обобщенный» или «агрегирующий» оператор в Access нет. Есть просто понятие «встроенные функции Microsoft Access», а среди них - «статистические функции» и «статистические функции по подмножеству».

Статистические функции - это: Sum (сумма), Count (количество записей, возвращаемых запросом), Avg (среднее), Var (дисперсия) и др., используемые для расчета итоговых значений. Статистическая функция, с помощью которой в запросе обрабатываются значения поля, может быть выбрана в ячейке строки "Групповая операция" в бланке запроса. Первоначально эта строка в бланке запроса отсутствует. Чтобы она появилась, надо выбрать позицию ***"Групповые операции"*** меню ***"Вид",*** или нажать кнопку со знаком "Σ" на панели инструментов.

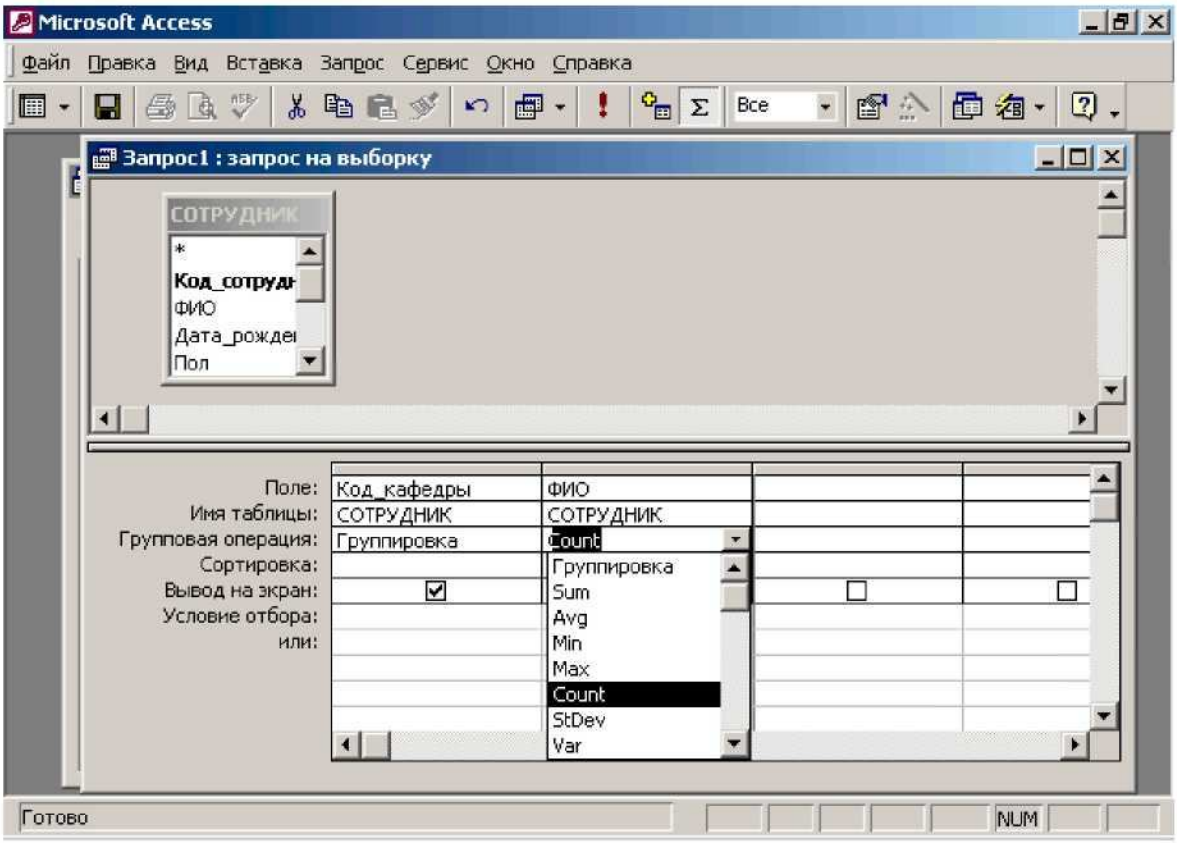


Рис. 2. 41. Использование групповых операций в запросах. Выбор агрегирующей функции

Первым полем, выводимым в ответ, должно быть поле, по которому производится группировка, а затем - поля, над которыми производятся вычисления. Все групповые операции, кроме Count, могут выполняться только над числовыми полями.

В строке **«Групповая операция»** щелчком мыши можно открыть список доступных функций, в котором можно осуществить выбор нужной статистической функций для выполняемых над полем вычислений.

На рис. 2.41 приведен пример использования групповых операций в запросе (Count - подсчет числа сотрудников, работающих на каждой кафедре).

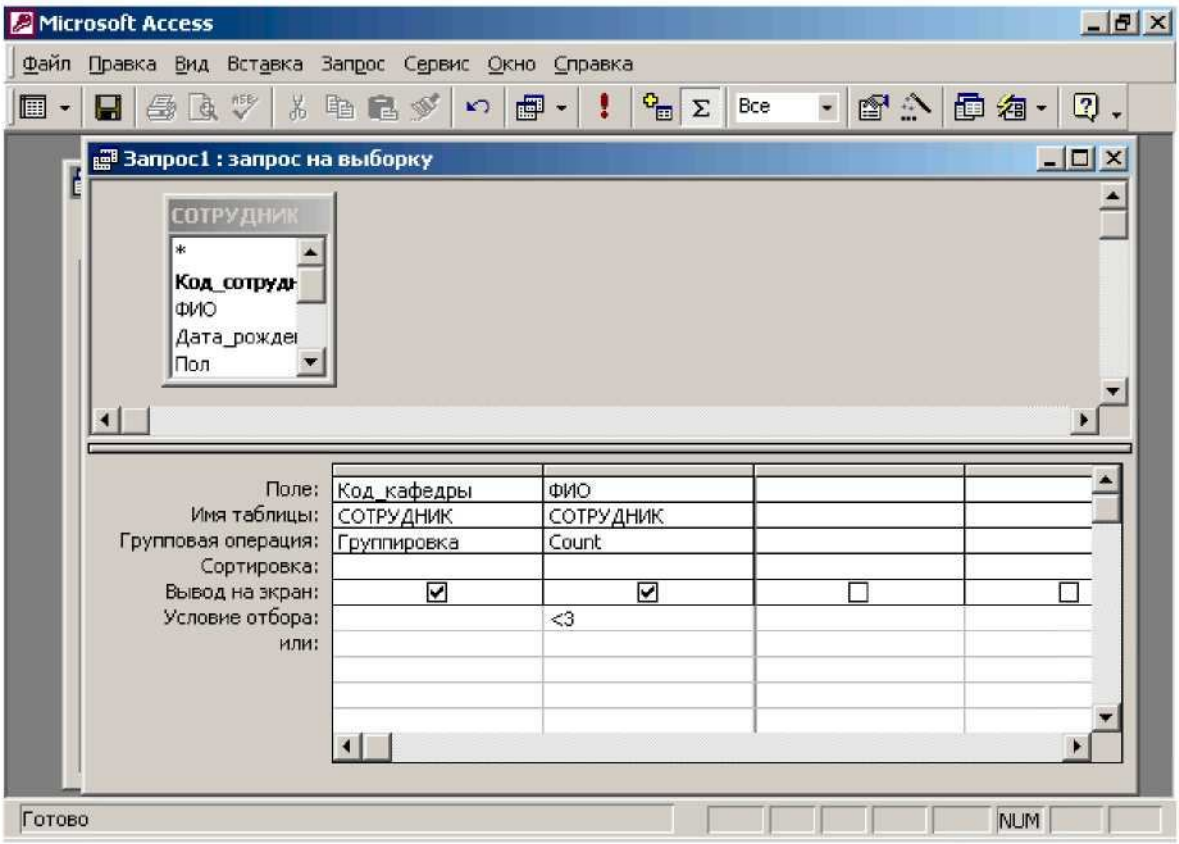


Рис. 2.42. Запрос с вычисляемым полем, используемым в условии отбора.

В Access предварительно упорядочивать таблицу по полю, по которому ведется группировка, не обязательно.

Выражения, определяющие вычисляемые поля, создаются с помощью мастера простых запросов или вводятся пользователем в строку *«Групповая операция»* бланка запроса. В бланке запроса задают также условия отбора, с помощью которых определяются группы, для которых вычисляются итоговые значения, записи, включаемые в вычисления, или результаты, отображаемые после выполнения расчетов. На рис. 2.42 изображен запрос, в котором условия отбора применены к вычисляемому полю («Выдать список кафедр, на которых работает меньше 3 человек»).

Если предположить (а это практически всегда так), что нет кафедр, на которых не работает ни одного человека, то результат запроса будет верен и когда задано внутреннее соединение, и если задано левое соединение. Но, предположим, что задается аналогичный по существу запрос «Выдать список сотрудников, имеющих меньше двух детей» на двух связанных таблицах «СОТРУДНИК» и «ДЕТИ». Всегда есть вероятность, что имеются сотрудники, которые не имеют детей. В случае если будет использовано внутреннее соединение (а оно задается по умолчанию), то такие сотрудники не попадут в ответ (т. е. результат ответа будет не соответствовать действительности). На результат запроса «Выдать список сотрудников, имеющих больше двух детей» параметры объединения таблиц не окажут влияния.

Запрос, использующий статистические функции, может не включать поле, по которому осуществляется группировка. В этом случае функция будет относиться ко всей совокупности отобранных данных.

Рассмотрим запросы, содержащие вычисляемые поля.

Существует ряд вычислений, которые можно выполнить в запросе, например, перемножить значения двух полей или вычислить дату, отстоящую на три месяца от текущей даты и т. п.

Выражения, определяемые пользователем, дают возможность выполнять действия с числами, датами и текстовыми значениями в каждой записи с использованием данных из одного или нескольких полей. Допустимые операции будут зависеть от типа полей, участвующих в выражении.

Так, для текстовых полей возможно использование оператора "+", который в этом случае воспринимается как конкатенация (соединение) строк.

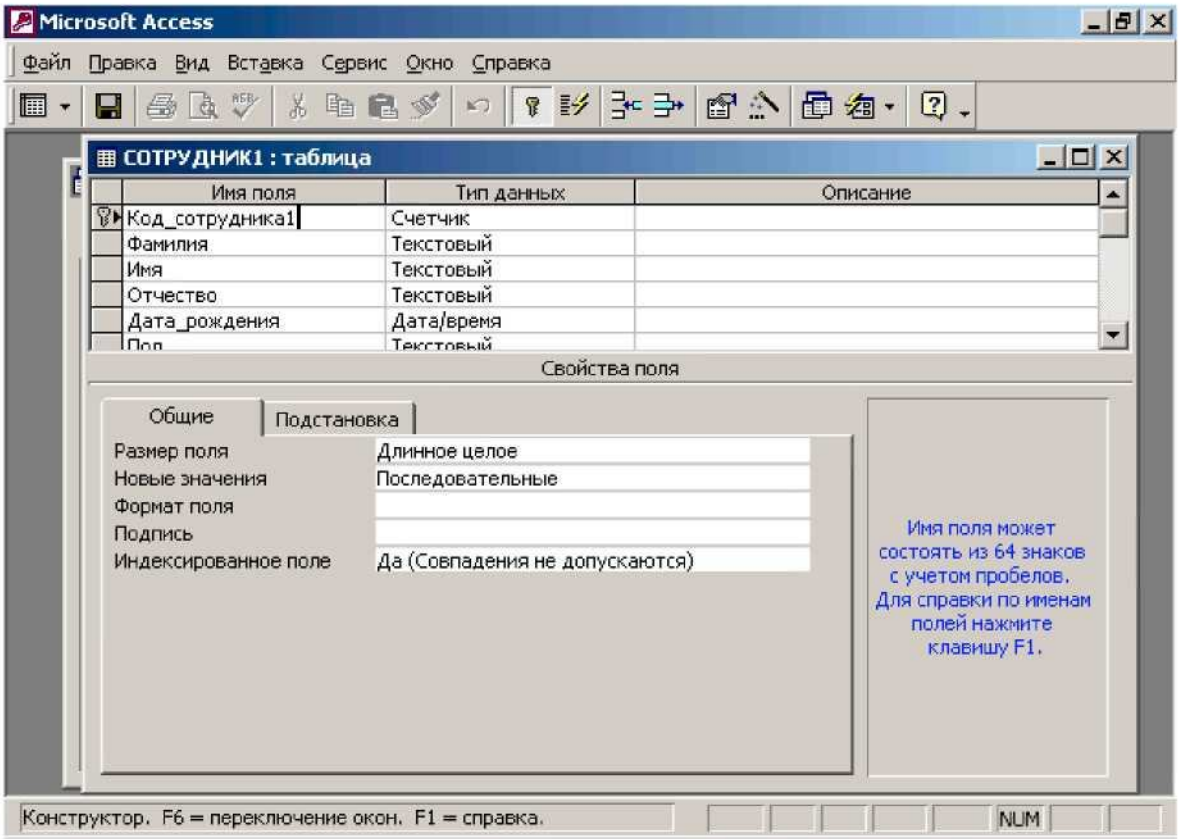


Рис. 2.43. Создание таблицы "СОТРУДНИК1"

Создадим таблицу "Сотрудник1" (рис. 2.43), подобную таблице "СОТРУДНИК", только поле "ФИО" разобьем на три поля: "Фамилия", "Имя", "Отчество". Предположим, что вы хотите вывести все три поля в одном столбце. Для этого можно использовать выражение:

[Фамилия] + " " + [Имя] + " " +[Отчество]

Если хотя бы одно из этих трех полей будет не заполнено (причем безразлично, будет это пустое поле, или там будут введены пробелы), то вся строка будет «пустой». В связи с этим рекомендуется вместо операции "+", использовать операцию "& ". В этом случае выражение будет иметь вид:

[СОТРУДНИК1]! [Фамилия] & " " & [СОТРУДНИК1]![Имя] & " " & [СОТРУДНИК1]! [Отчество]

Для расчетов с использованием формул, определяемых пользователем, требуется создать новое вычисляемое поле прямо в бланке запроса. Вычисляемое поле создается путем ввода требуемого выражения в пустую колонку в строку «поле» в бланке запроса (рис. 2.44).

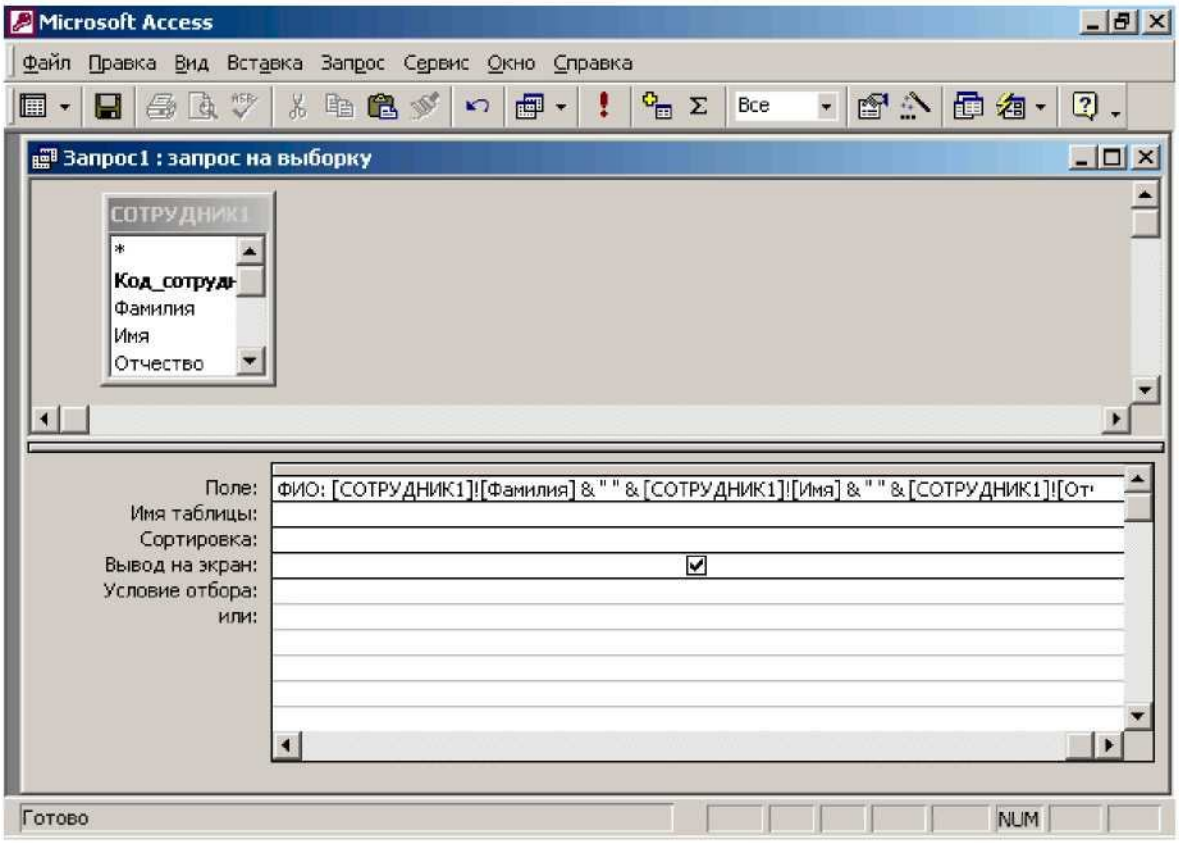


Рис. 2.44. Использование выражений с тестовыми полями

Выражение может вводиться вручную, а можно для этих целей воспользоваться построителем выражений.

Для получения списка, включающего фамилии и инициалы сотрудников можно использовать следующее выражение:

фамилия_инициалы: [СОТРУДНИК1]![Фамилия]&" " & Left([СОТРУДНИК1]![Имя];1) & "." & " " & Left([СОТРУДНИК1]![Отчество];1) & "."

В этом выражении использована функция Left. Второй аргумент этой функции определяет число возвращаемых символов.

Для числовых значений можно использовать любые арифметические операторы. На рис. 2.45 используется операция сложения над числовыми величинами.

Результаты вычислений не обязательно должны отображаться в ответе. Их можно использовать в условиях отбора для определения записей, которые выбираются в запросе, или для определения записей, над которыми производятся какие-либо действия. Например, на рис. 2.45 изображен запрос: «Выдать список сотрудников, зарплата которых превышает 2000 рублей». Само вычисляемое поле только используется в условиях отбора, но в ответ не выводится.

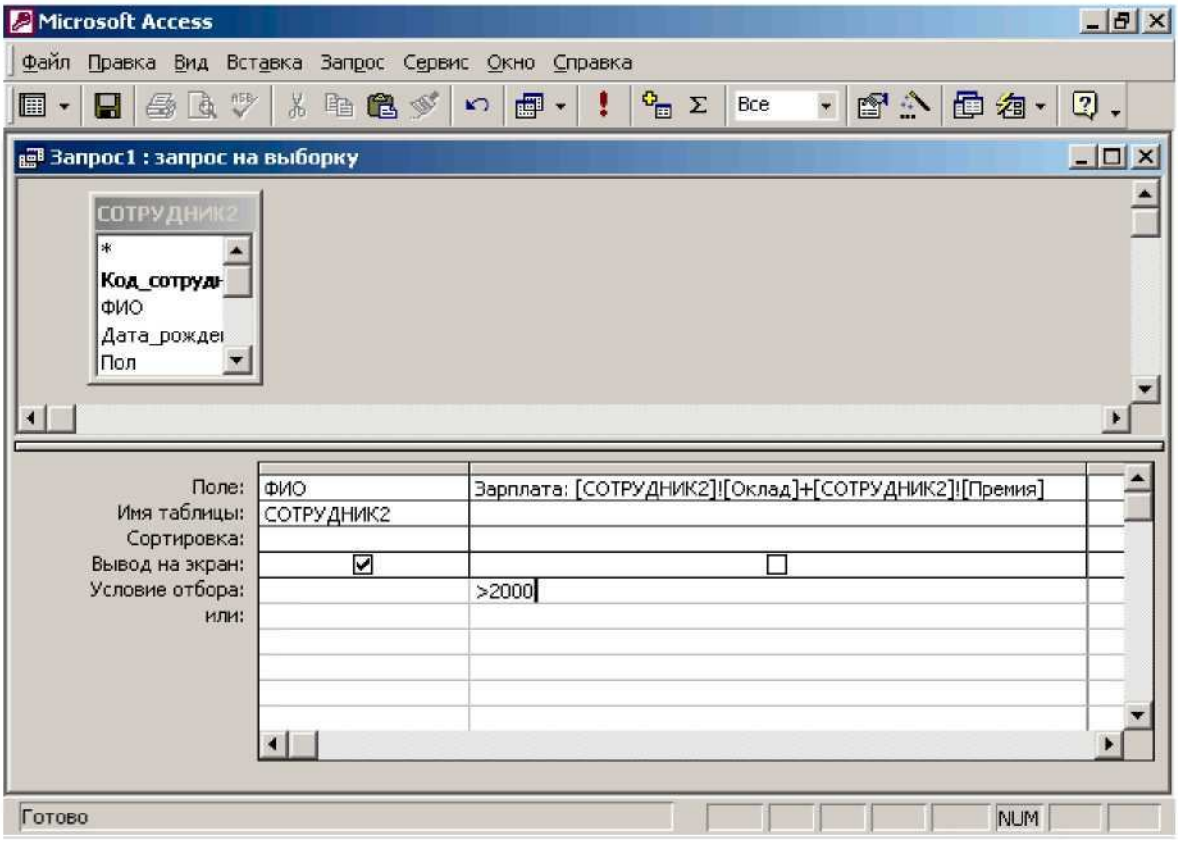


Рис. 2.45. Запрос с вычисляемым полем, используемым в условии отбора

Предполагается, что зарплата состоит из оклада и фиксированной премии/надбавки. Для создания этого запроса скорректируйте таблицу «СОТРУДНИК», добавив в нее поле «ПРЕМИЯ» (рис. 2.46). Введите в это поле данные.

Следует обратить внимание на операции над датами. Над полями с данным типом можно производить следующие действия:

- от даты можно отнять другую дату; при этом получается число, показывающее, на сколько дней отстоит одна дата от другой. Если вы хотите получить интервал времени в других единицах измерения, то следует воспользоваться функцией DateDiff.
- от даты можно отнять/прибавить число; при этом получается дата, отстоящая от данной на заданное число дней.

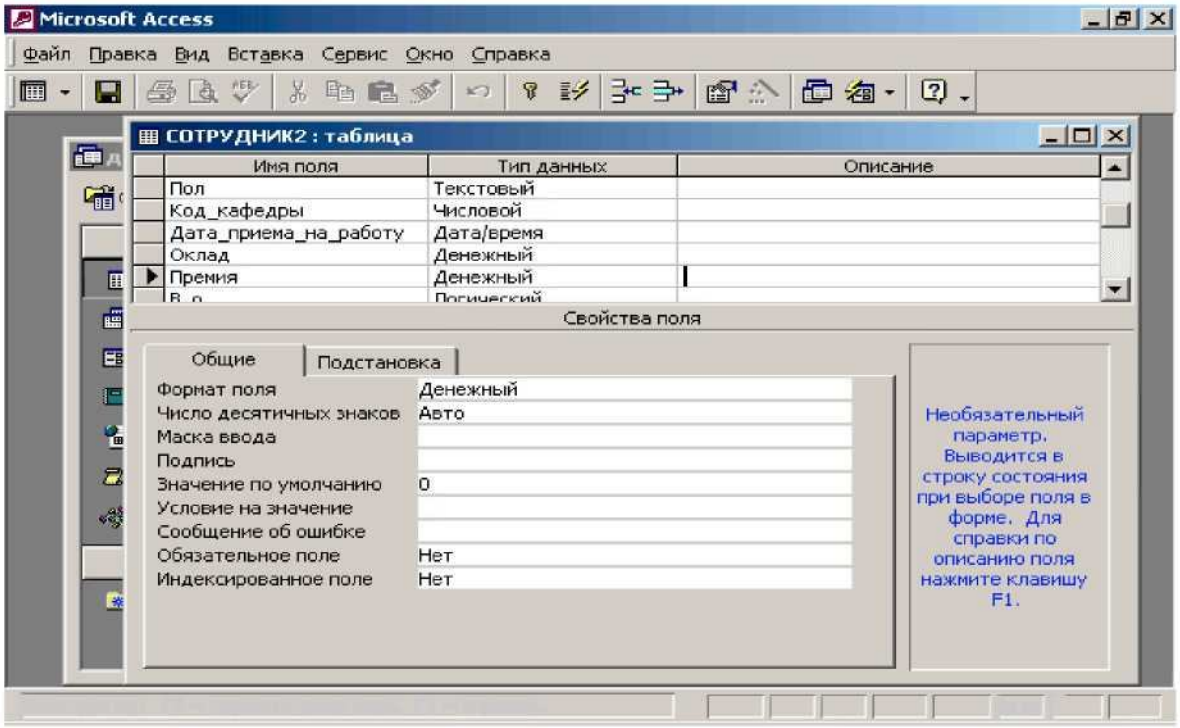


Рис. 2.46. Фрагмент измененной структуры таблицы «Сотрудник»

Рассмотрим перекрестные запросы.

Перекрестные запросы служат для более компактного отображения информации (схожего с изображением в электронных таблицах). В перекрестном запросе отображаются результаты статистических расчетов (такие как: суммы, количество записей, средние значения), выполненных по данным из одного поля. Эти результаты группируются по двум наборам данных в формате перекрестной таблицы. Первый набор выводится в левом столбце и образует заголовки строк, а второй выводится в верхней строке и образует заголовки столбцов.

Например, в таблице «СОТРУДНИК» имеются сведения об окладе каждого сотрудника, а также признаки, на какой кафедре и в какой должности работает каждый сотрудник. Требуется для каждой кафедры определить общий фонд зарплаты, а по каждой должности - среднюю по каждой кафедре зарплату.

Для создания перекрестного запроса следует воспользоваться позицией **«Перекрестный запрос»** в окне **«Новый запрос»** (рис. 2.25) или выбрать соответствующую строку в меню **«Запрос».** Далее надо выполнить ряд шагов, предлагаемых мастером по созданию перекрестных запросов (рис. 2.47-2.50). Вид запроса, полученного в результате использования мастера, представлен на рис. 2.50

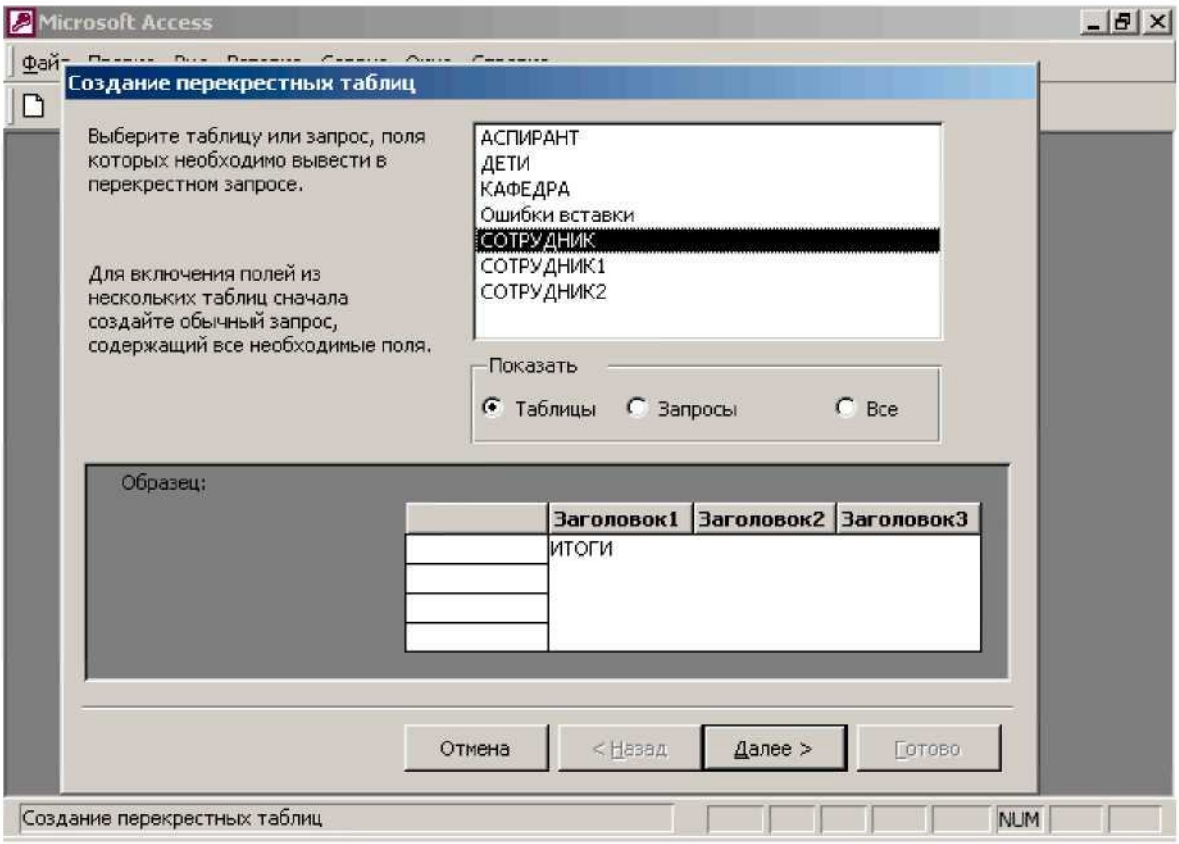


Рис.2.47. Создание перекрестного запроса (шаг 1)

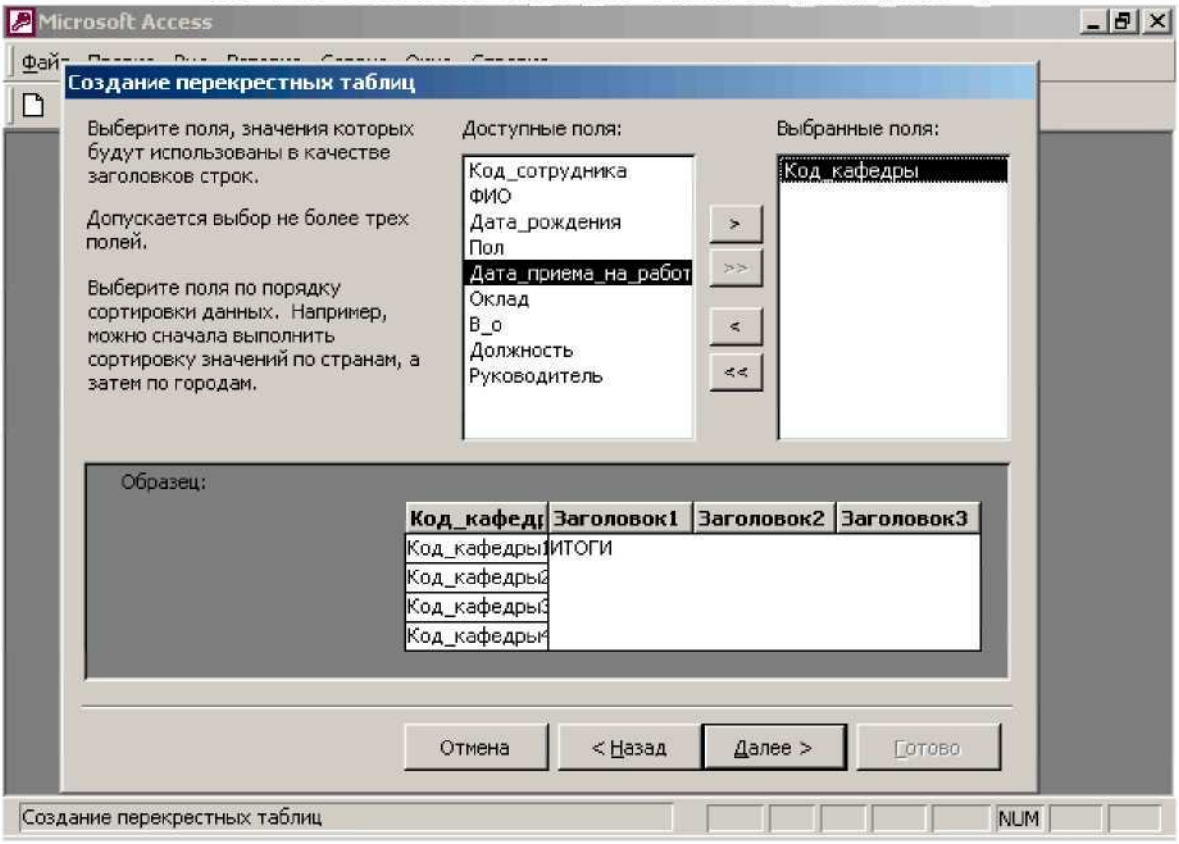


Рис.2.48. Создание перекрестного запроса (шаг 2)

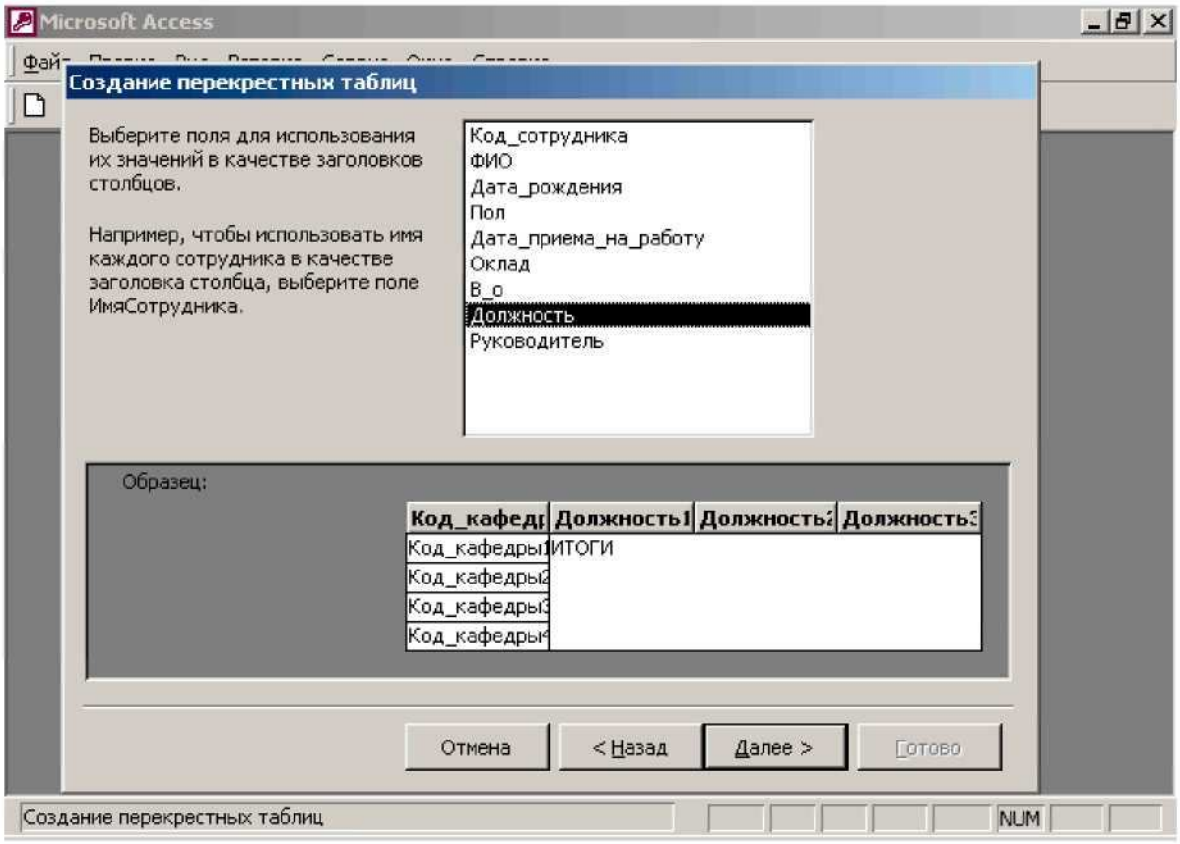


Рис. 2.49. Создание перекрестного запроса (шаг 3)

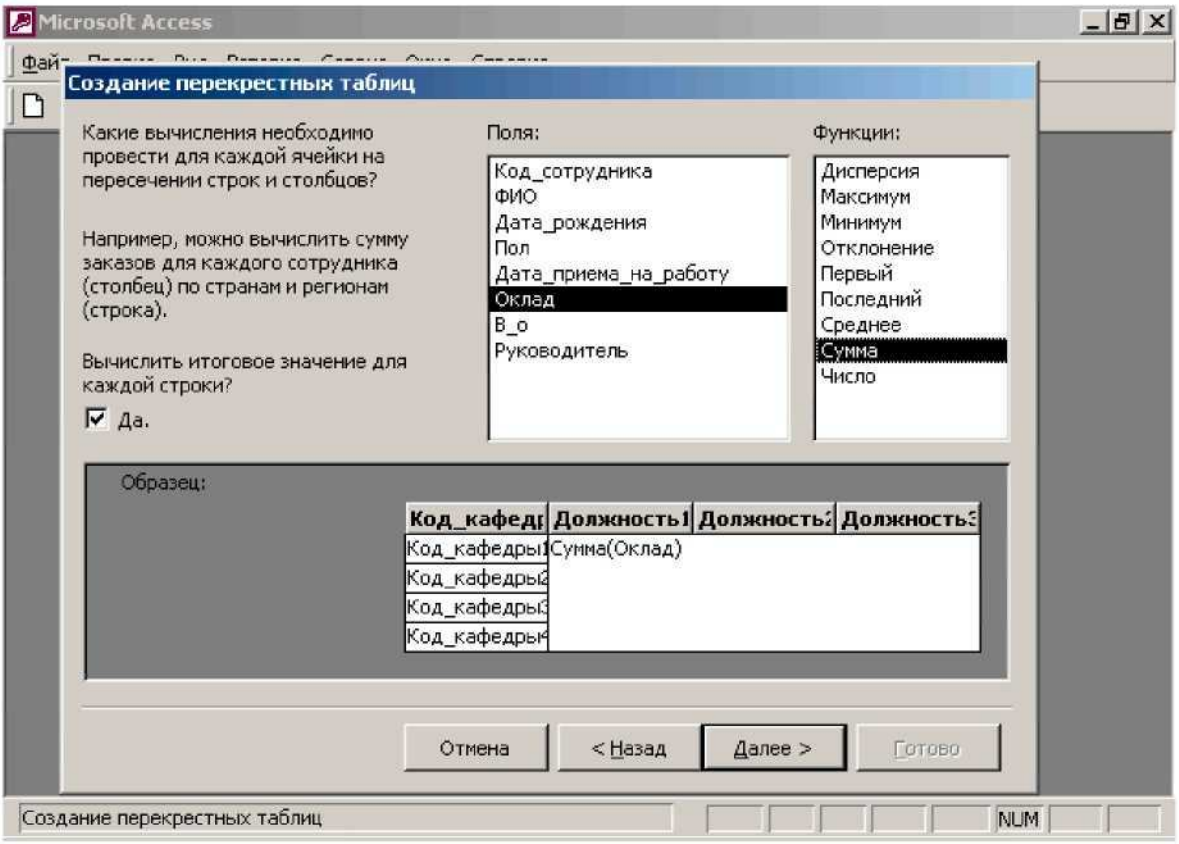


Рис.2.50. Перекрестный запрос (1)

Этот запрос не совсем соответствует тому, что мы хотели получить в ответе, так как в результате для каждой кафедры будет посчитана средняя зарплата, а не общий фонд, как требовалось. Для того чтобы получить требуемую сумму, надо сделать соответствующее изменение в конструкторе запросов (рис. 2.51):

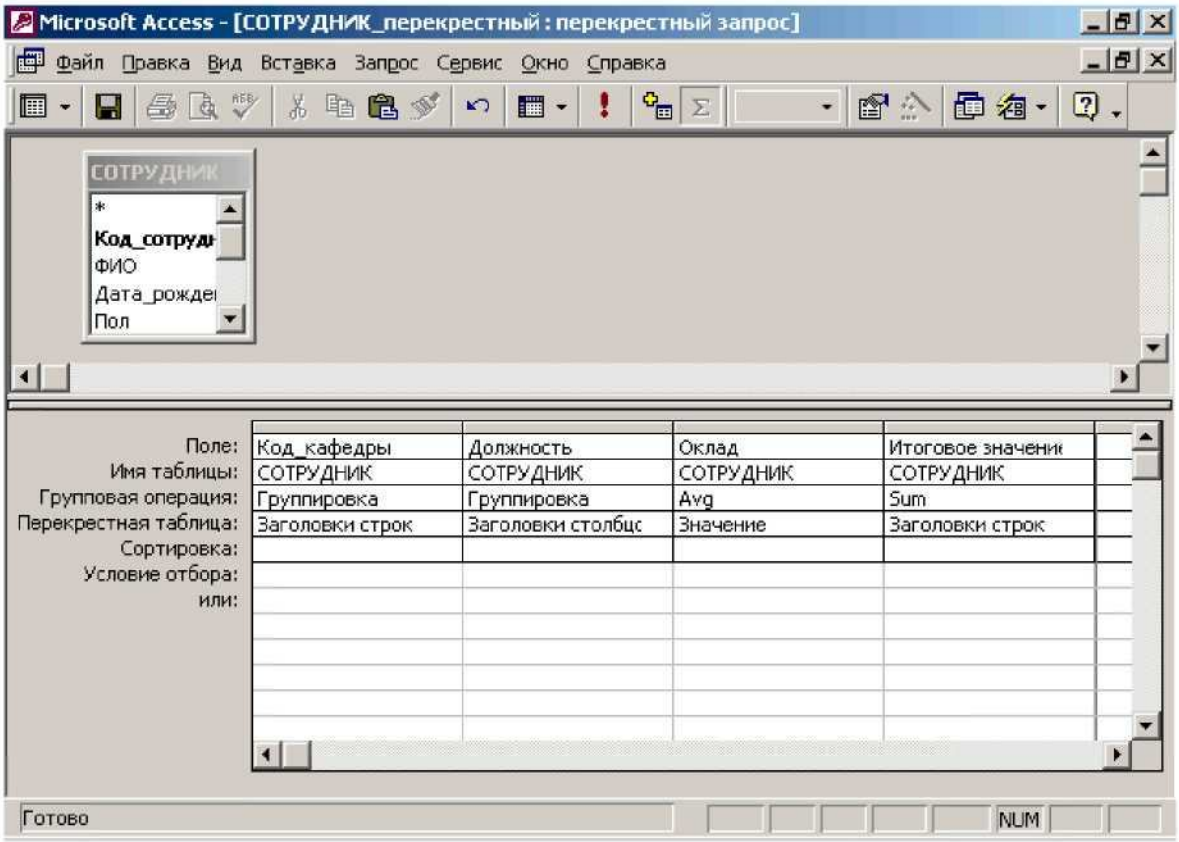


Рис. 2.51. Перекрестный запрос(2)

Запросы с параметрами

Если приходится часто выполнять однотипный запрос на выборку или перекрестный запрос, изменяя при этом значение какого-либо атрибута в условии отбора, то можно использовать запрос с параметрами. Запрос с параметрами не требует каждый раз вносить изменения в бланк запроса; вместо этого выводится приглашение пользователю ввести условия отбора. Запрос с параметрами особенно удобно использовать, когда конечные пользователи, для которых создается система, не очень квалифицированные, и желательно, чтобы они сами никакие элементы системы не изменяли. Запросы с параметрами также удобно использовать в качестве основы для форм и отчетов. Например, на основе запроса с параметрами можно создавать ежемесячные отчеты. При выводе такого отчета, на экране появится приглашение ввести месяц, за который должен быть получен отчет.

Для каждого поля, которое предполагается использовать как параметр, в конструкторе запросов надо ввести в ячейку строки **«Условие отбора»** текст приглашения, заключенный в квадратные скобки. Это приглашение будет выводиться при запуске запроса. Текст подсказки должен отличаться от имени поля, но может включать его. На рис. 2.52 представлен параметрический запрос для получения списка сотрудников заданной кафедры.

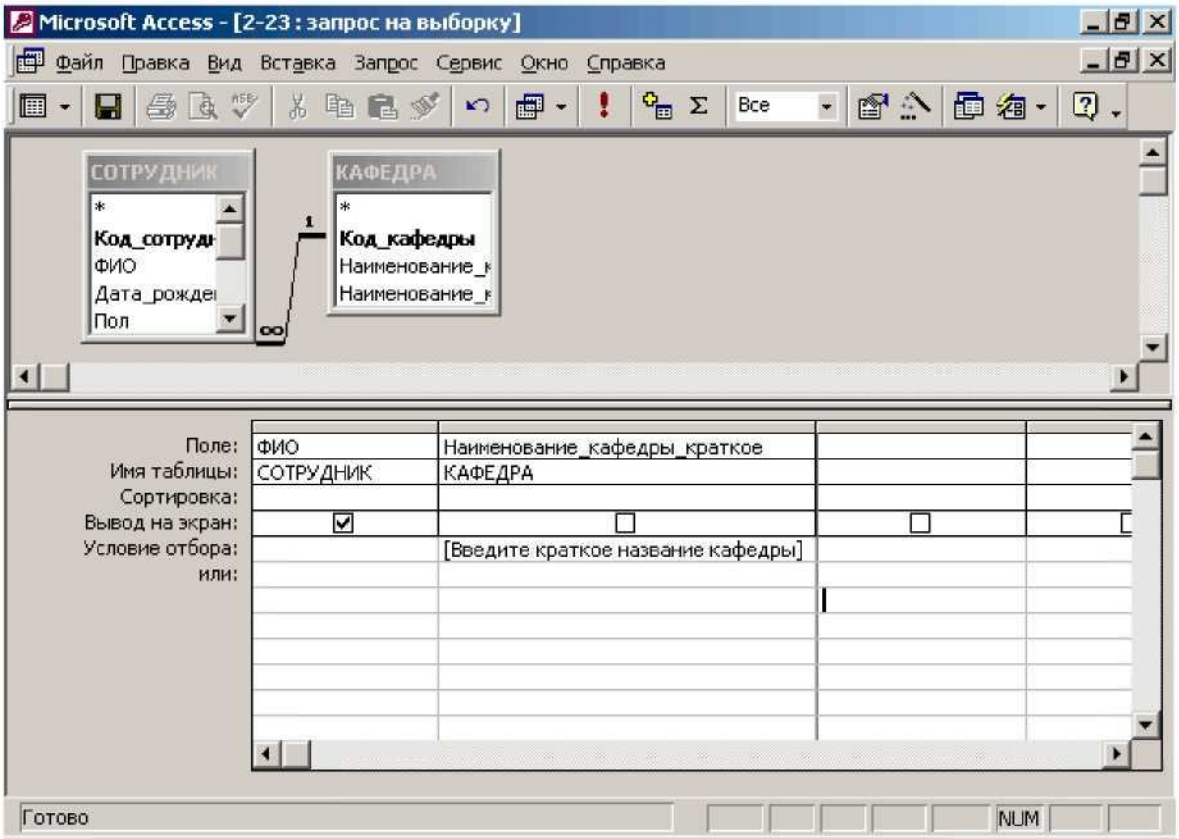


Рис. 2.52. Параметрический запрос

При запуске этого запроса на исполнение будет появляться окно с подсказкой, в которое надо ввести требуемое значение параметра.Для каждого поля можно задать не только одно конкретное значение, но и диапазон значений. Так, например, для поля, в котором отображаются даты, можно вывести приглашения «Введите начальную дату:» и «Введите конечную дату:» для определения диапазона отбираемых значений. Для этого в соответствующую ячейку строки «Условие отбора» надо ввести выражение **Between** [Введите начальную дату:] **And** [Введите конечную дату:].

В качестве параметров может быть использовано не одно, а несколько полей. В этом случае для каждого поля, которое предполагается использовать как параметр, в ячейку строки **«Условие отбора»** вводится текст приглашения, заключенное в квадратные скобки. Эти приглашения будут последовательно выводиться при запуске запроса.

Корректирующие запросы

Корректирующие запросы *(запрос на обновление* (Update), *удаление* (Delete), *добавление (Apptnd))* могут изменять как все записи таблицы, так и определенное их подмножество - это будет зависеть от условия отбора.

Для создания корректирующего запроса надо в режиме конструктора запроса выбрать соответствующую позицию в меню **«Запрос»** (либо нажать стрелку рядом с кнопкой **«Тип запроса»** на панели инструментов), как показано на рис. 2.53.

Чтобы просмотреть обновляемые записи перед выполнением запроса можно нажать кнопку **«Вид»** на панели инструментов. Выводимый список будет содержать старые значения полей отобранных в запросе записей.

Надо быть очень внимательным перед выполнением корректирующих запросов (не даром в списке запросов перед их именем стоит восклицательный знак), так как каждый их запуск на выполнение изменяет содержимое таблиц, и отменить результат выполнения нельзя. Перед выполнением корректирующего запроса рекомендуется сохранить копию изменяемых таблиц.

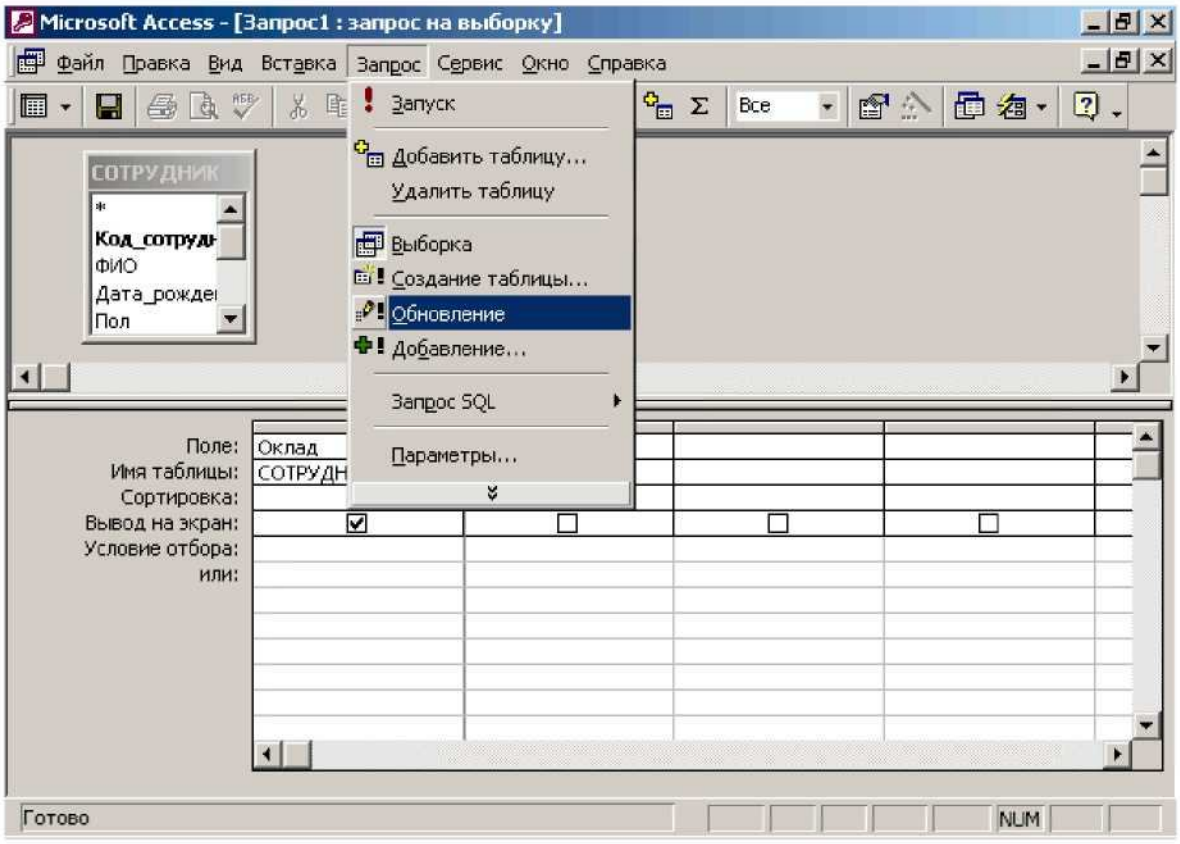


Рис. 2.53. Создание запроса на обновление

Запрос на обновление

Запрос, изображенный на рис. 2.54, увеличит зарплату всех сотрудников (так как условие отбора не задано) на 30%.

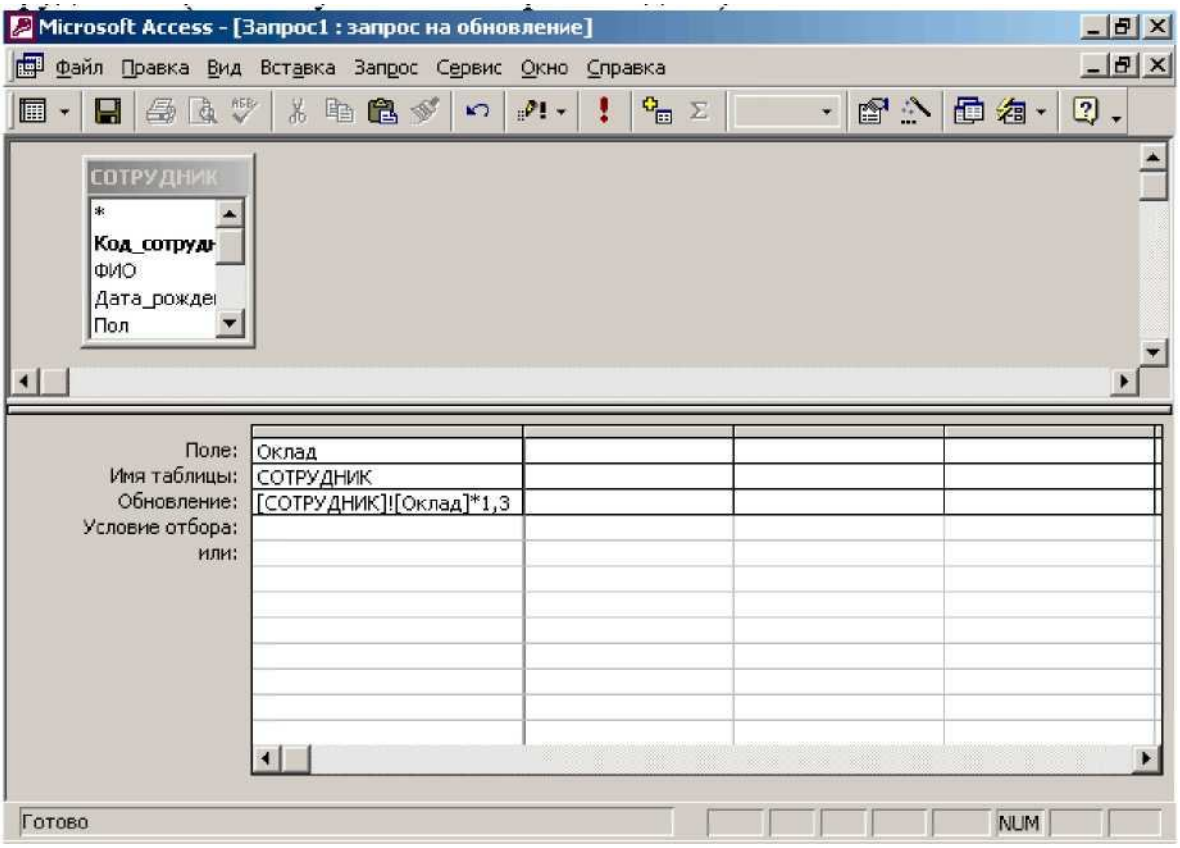


Рис. 2.54. Запрос на обновление(1)

Запрос, изображенный на рис. 2.55, изменит зарплату одного конкретного сотрудника (см. условие отбора данного запроса) и установит для нее значение, указанное в запросе.

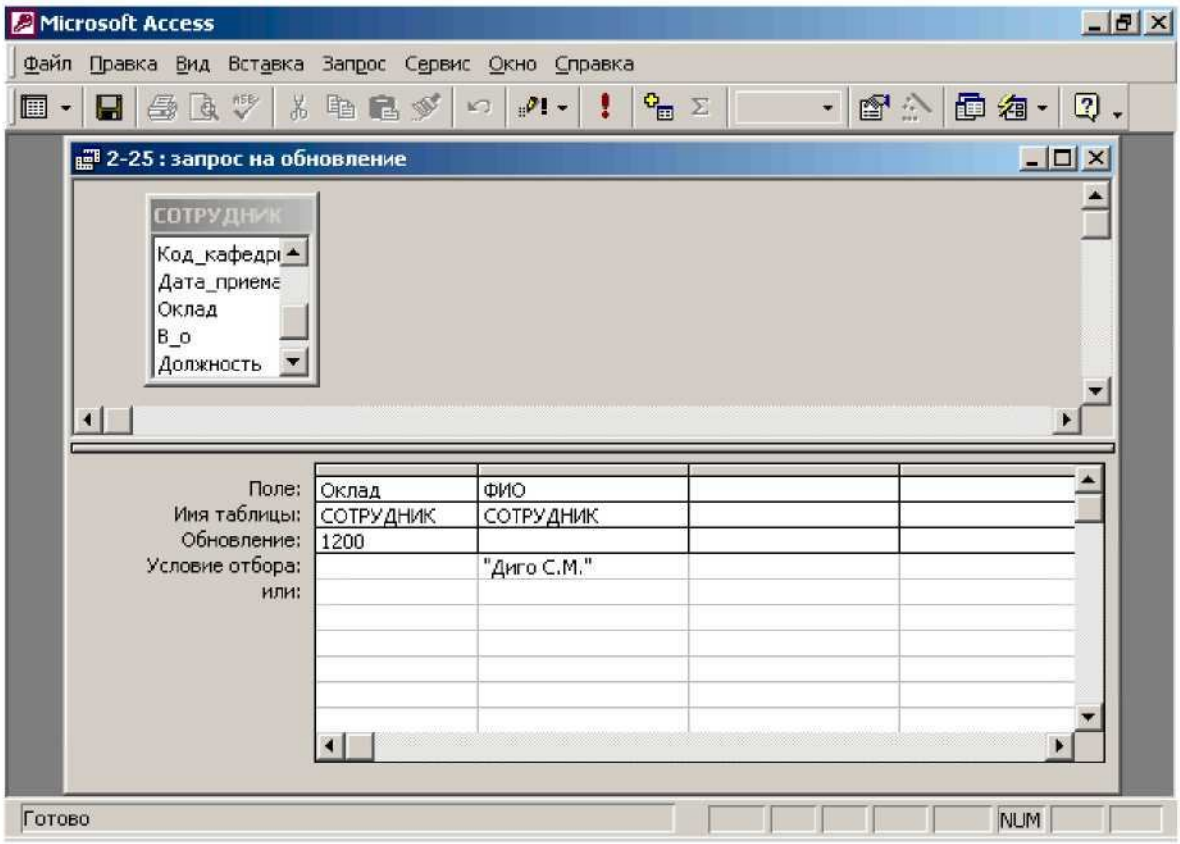


Рис. 2.55. Запрос на обновление (2*)*

При выполнении *корректирующих запросов* система осуществляет контроль ограничений целостности. Так, например, если при описании таблицы было задано ограничение на максимально допустимое значение поля «Оклад», то при выполнении запроса, изображенного на рис. 2.54, в случае нарушения ограничения может быть выдано сообщение типа изображенного на рис. 2.56. Если Вы выберите вариант «Да», то для записей, нарушающих ограничение на значение, корректировка вообще выполнена не будет. В нашем примере такой выбор будет неправильным, так как в этом случае, у одних сотрудников зарплата будет повышена, а у других - нет. Причем потом разобраться, у кого она была повышена, а у кого - нет, будет практически невозможно. В рассматриваемой ситуации следует отказаться от выполнения запроса, скорректировать при необходимости ограничение на значение и только после этого выполнить запрос.

Рис. 2.56. Контроль ограничений целостности при выполнении корректирующих запросов

Запрос на удаление

При выборе «запроса на удаление» в бланке запроса в строке «Условие отбора», также как и в запросах на выборку, задается условие. Записи, удовлетворяющие условию, будут удалены из базы данных. Для того, чтобы быть уверенным, что запрос задан верно, и, как следствие, удаляться именно те записи, которые необходимо, рекомендуется сначала задать запрос на выборку, посмотреть полученный результат, а затем изменить тип, запроса, выбрав «запрос на удаление». Если в «запросе на удаление» не задано никаких условий отбора, то из таблицы удалятся все записи.

Запрос на добавление

Запрос на добавление добавляет группу записей из одной или нескольких таблиц в конец одной или нескольких таблиц).

Для задания запроса такого типа надо сначала создать запрос, содержащий таблицу, записи из которой необходимо добавить в другую таблицу. Затем в режиме конструктора запроса надо нажать стрелку рядом с кнопкой **«Тип запроса»** на панели инструментов и выбрать команду «Добавление» (либо выбрать соответствующую позицию в меню **«Запросы»).** На экране появится диалоговое окно "Добавление". В поле "ИМЯ ТАБЛИЦЫ" надо ввести имя таблицы, в которую необходимо добавить записи.

Таблица, в которую осуществляется добавление, может быть как в той же базе данных, так и в другой, причем это не обязательно должна быть база данных Access (это может быть Microsoft FoxPro, Paradox или dBASE, а также б аза данных SQL).

Из списка полей в бланк запроса надо переместить поля, которые необходимо добавить, а также те, которые будут использованы при определении условия отбора.

Если все поля в обеих таблицах имеют одинаковые имена, то можно просто переместить с помощью мыши символ «звездочка» (*) в бланк запроса. Однако при работе с репликой базы данных добавлять придется все поля. Кроме того, при использовании символа «звездочка» (*), даже если структуры обеих таблиц полностью совпадают, могут возникнуть проблемы с ключами (если ключевое поле имеет тип счетчик, то для автоматического добавления значений счетчика не следует при создании запроса перемещать поле счетчика в бланк запроса).

Если в обеих таблицах выделенные поля имеют одинаковые имена, соответствующие имена автоматически вводятся в строку **«Добавление».** Если имена полей двух таблиц отличны друг от друга, в строку **«Добавление»** надо ввести имена полей, добавляемых в таблицу.

Запрос на создание таблицы

Запрос на создание таблицы фактически означает запоминание результата запроса в таблице. Чтобы использовать такую возможность надо создать запрос, результат которого следует поместить в новую таблицу. Затем в режиме конструктора запроса надо выбрать **«Тип запроса» - «Создание таблицы»** (рис. 2.57).

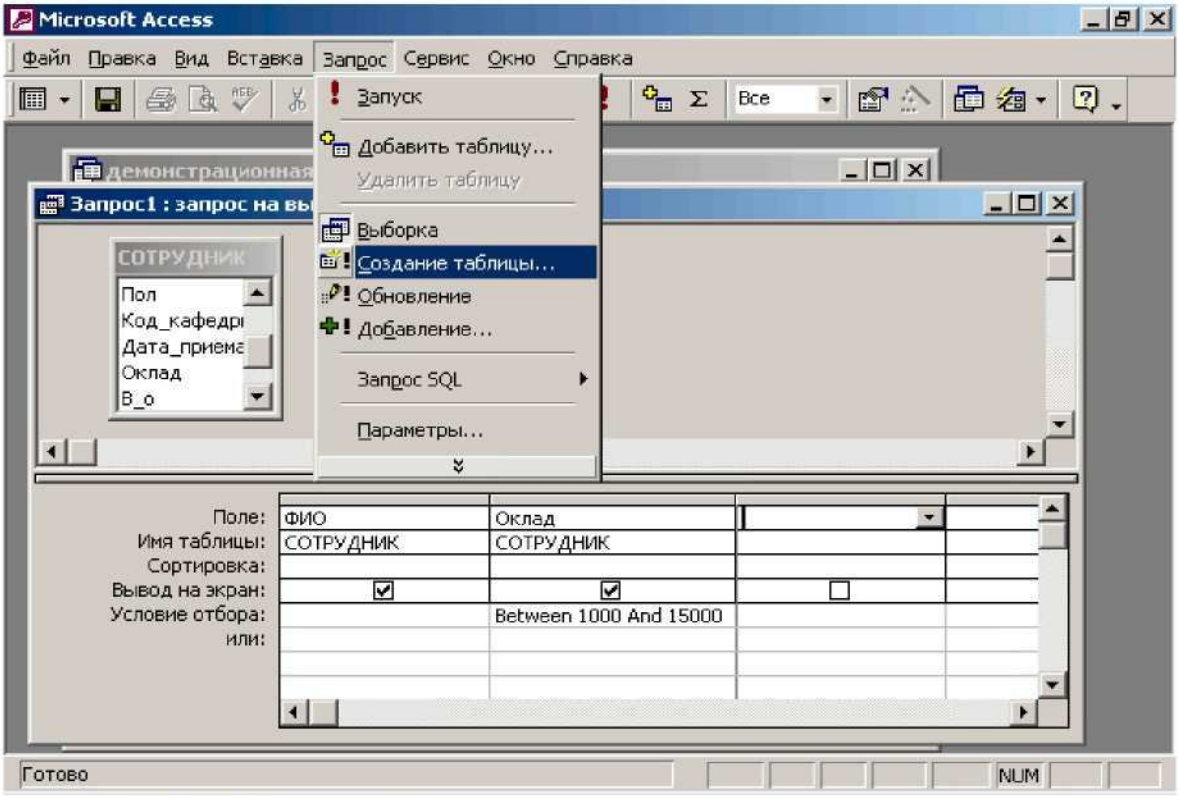


Рис. 2.57. Создание таблицы путем запоминания результата запроса (экран 1)

На экране появится диалоговое окно «Создание таблицы» (рис.2.58).

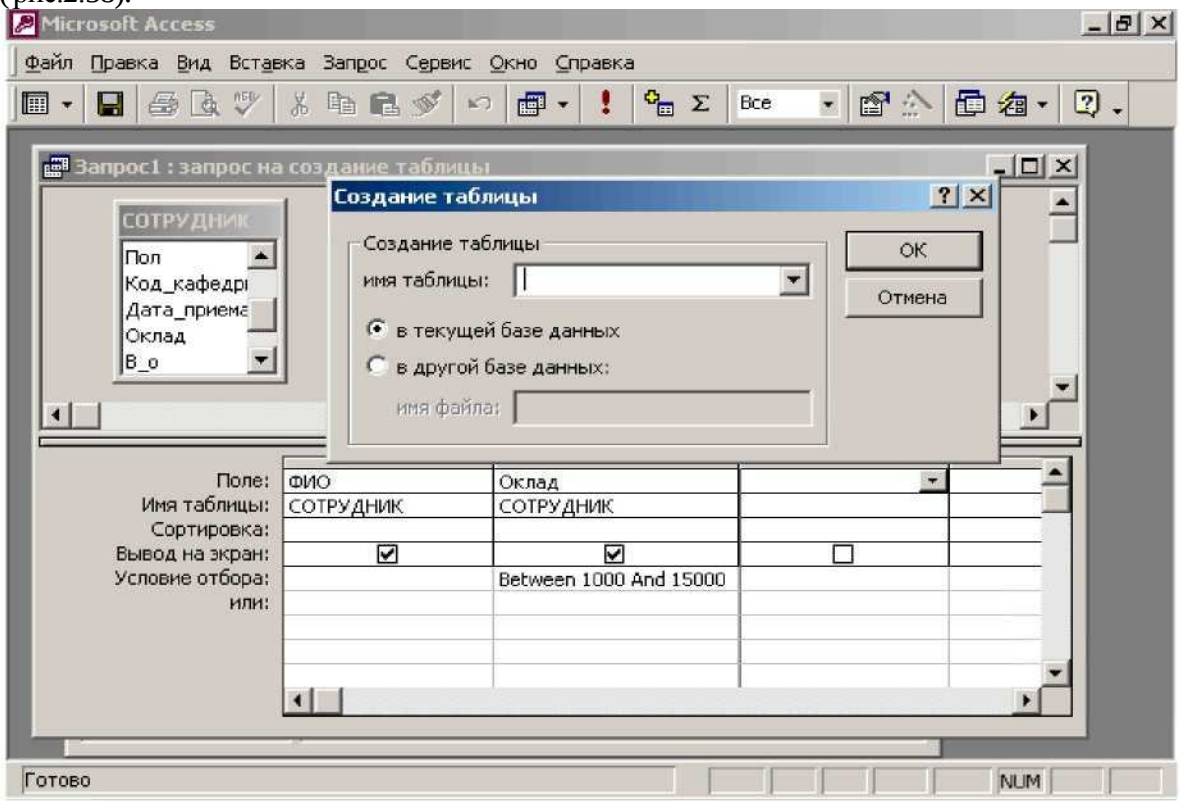


Рис. 2.58. Создание таблицы путем запоминания результата запроса (экран 2)

В поле «ИМЯ ТАБЛИЦЫ» надо ввести имя таблицы, в которую будут переноситься данные.

Дополнительные возможности

В Access при задании запросов можно использовать дополнительные возможности, которые упрощают задание запросов некоторых видов.

Поиск записей, не имеющих подчиненных

Необходимость поиска записей, не имеющих подчиненных, возникает довольно-таки часто и не только для проверки целостности базы данных. В нашем примере мы воспользуемся такой возможностью для определения списка сотрудников, не имеющих детей.

Для того чтобы воспользоваться возможностью поиска записей, не имеющих подчиненных, можно выбрать мастер **«Записи без подчиненных»** в окне **«Новый запрос» (**см. рис. 2.59**).**

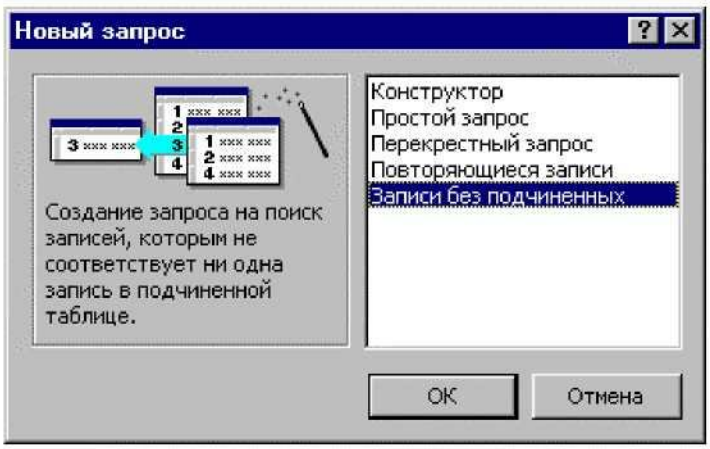


Рис. 2.59. Выбор мастера «Записи без подчиненных»

Затем надо выбрать основную таблицу (рис. 2.60) в паре «основная - подчиненная». В нашем случае это таблица «СОТРУДНИК». Основная и подчиненная таблицы должны быть предварительно связаны в схеме данных.

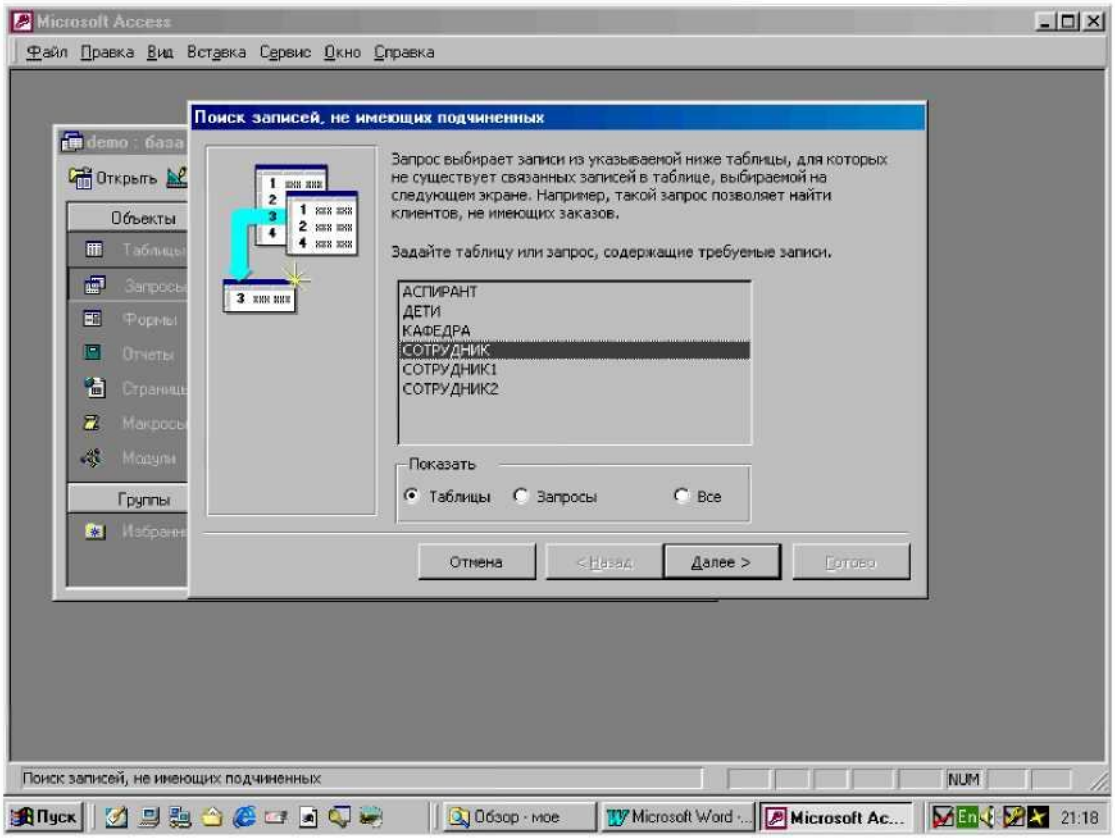


Рис. 2.60. Поиск записей, не имеющих подчиненных. Шаг 1. Выбор основной таблицы

Далее выбирается подчиненная таблица. В нашем случае это таблица «ДЕТИ» (рис. 2.61).

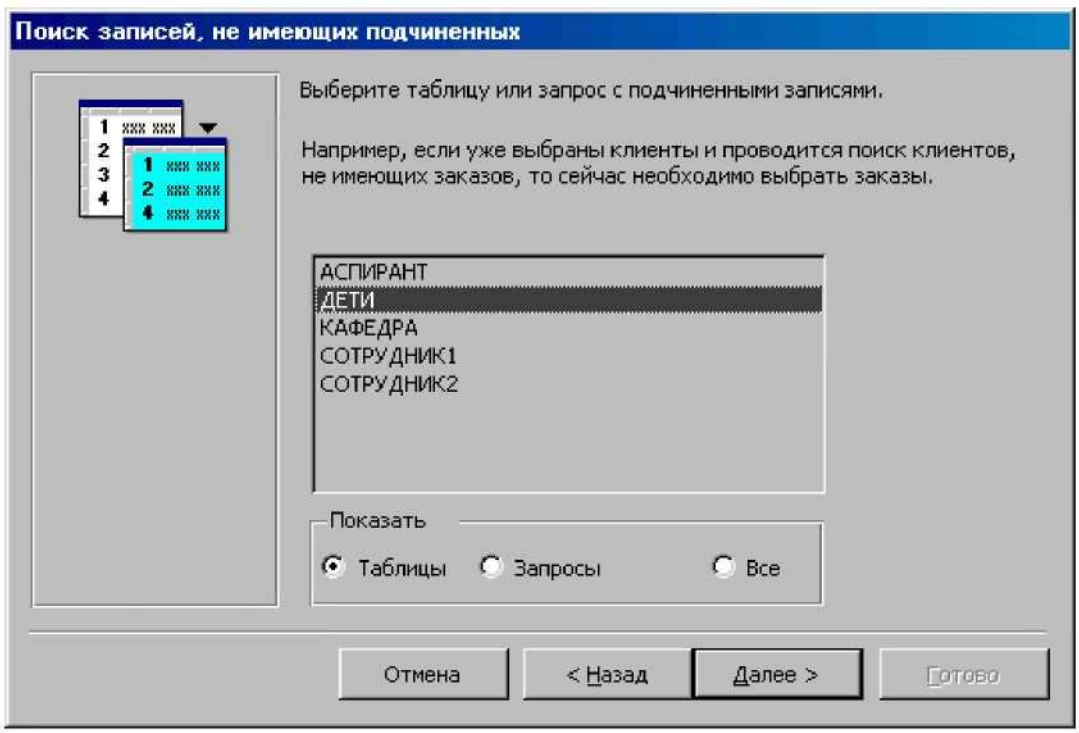


Рис. 2.61. Поиск записей, не имеющих подчиненных. Шаг 2. Выбор подчиненной таблицы

На следующем шаге определяются поля, по которым связаны выбранные таблицы (рис. 2.62).

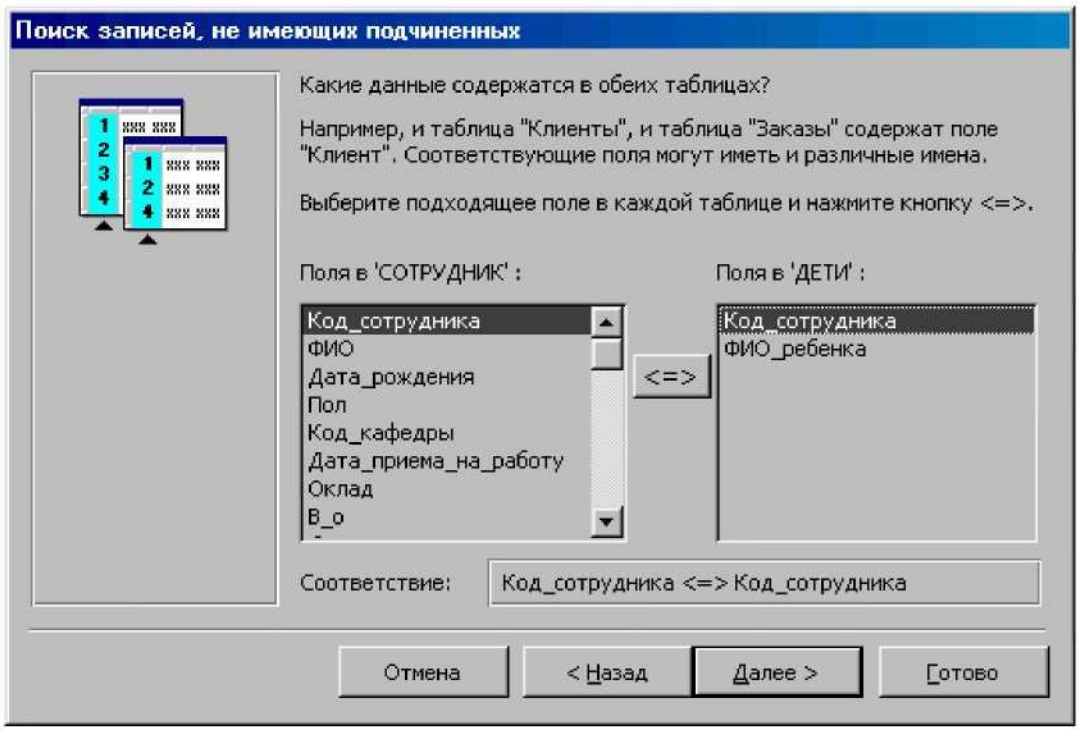


Рис. 2.62. Поиск записей, не имеющих подчиненных. Шаг 3. Определение полей связи

Затем выбираются поля, которые должны войти в ответ (рис. 2.63). Так как необходим просто список сотрудников, то в ответ выводится только поле «ФИО».

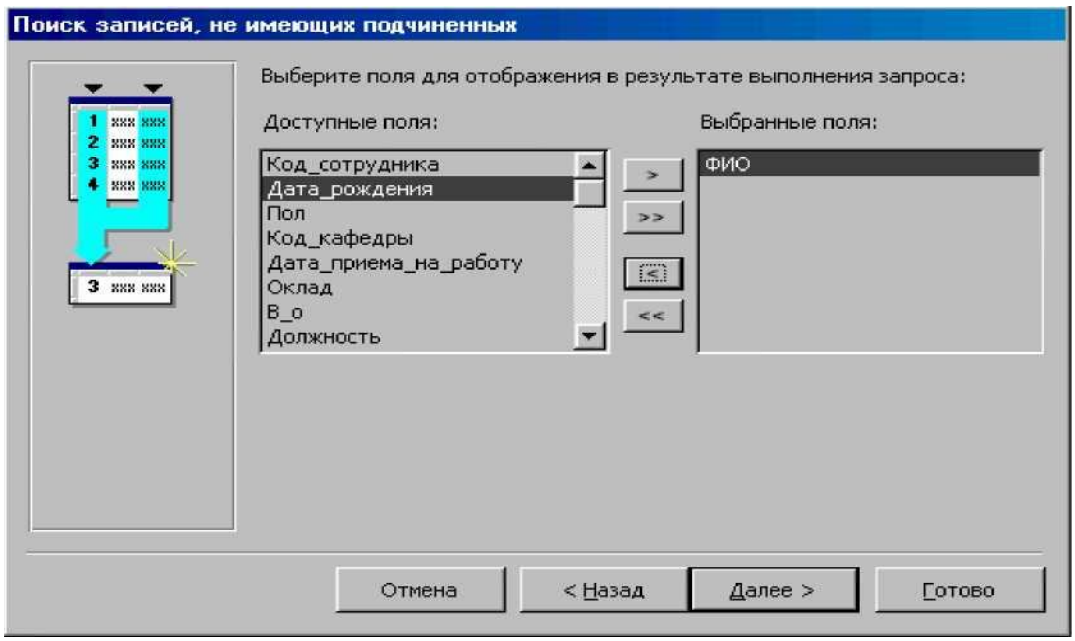


Рис. 2.63. Поиск записей, не имеющих подчиненных. Шаг 4. Определение полей, включаемых в ответ

В завершении надо задать имя созданного запроса (рис. 2.64).

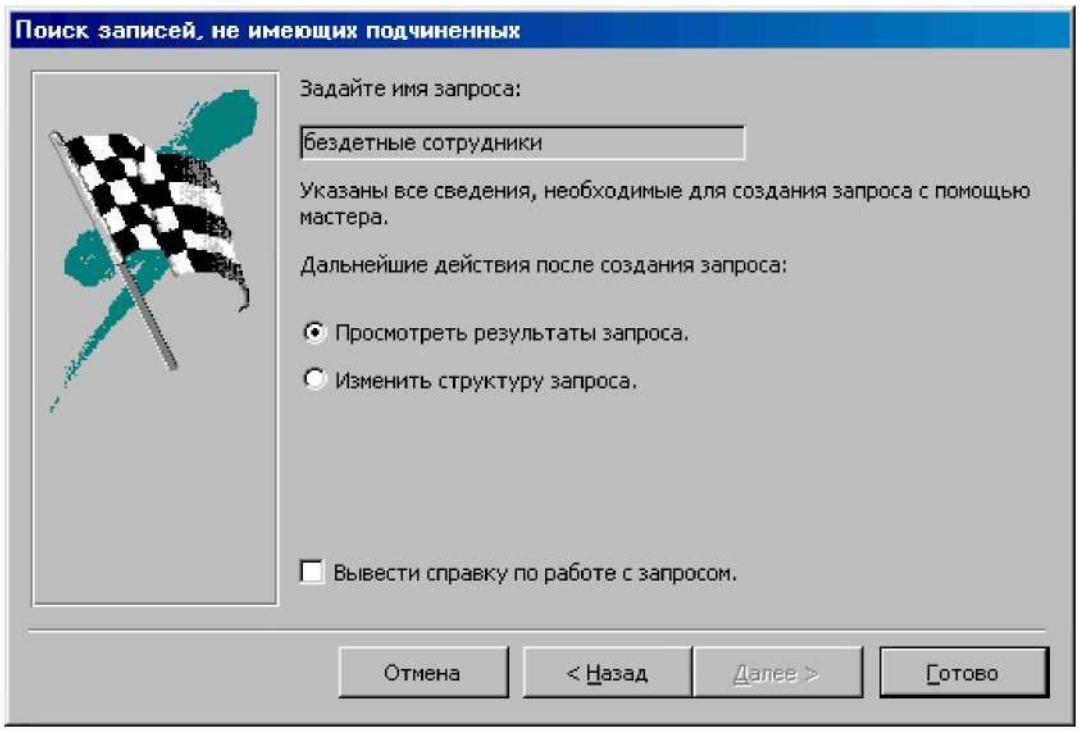


Рис. 2.64. Поиск записей, не имеющих подчиненных. Шаг 5. Задание имени запроса

Рассматриваемый запрос можно было задать и не пользуясь мастером. Посмотрим, как выглядит созданный нами запрос в режиме конструктора (рис. 2.65). Создание подобных запросов и в режиме конструктора не представляет особых трудностей. Но надо обратить внимание, что при связи основной и зависимой таблицы обязательно должно быть определено так называемое «левое соединение» (т. е. для связи в «параметрах объединения» надо выбрать вторую возможность - «объединение всех записей из первой таблицы и только тех записей из второй таблицы, в которых связанные поля совпадают). В противном случае список окажется пустым.

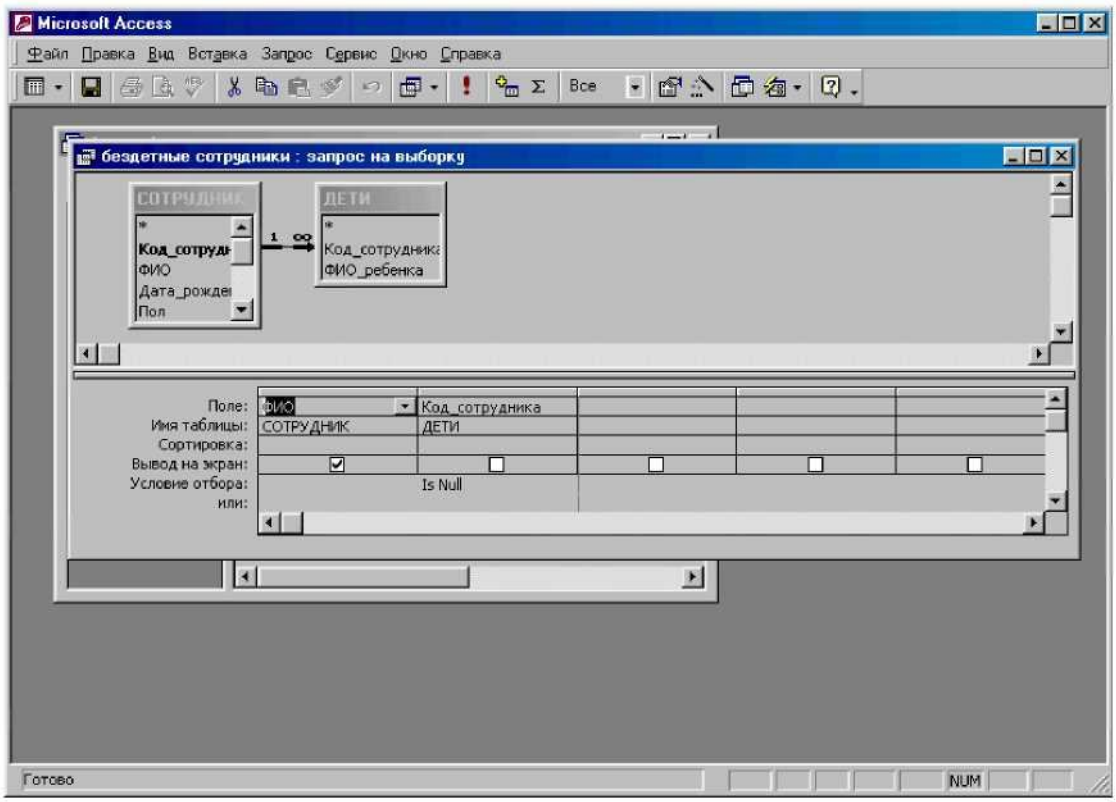


Рис. 2.65. Вид запроса в режиме конструктора.

Определение числа записей, выводимых в ответ

По умолчанию в ответ выводятся все отобранные записи. В Access есть возможность управлять числом записей, выводимых в ответ. Эта возможность выходит за рамки двухмерных табличных языков. Указанной возможностью можно пользоваться не только для ограничения числа записей, если отобранное множество слишком велико и для пользователя является приемлемым ограничиться определенным его подмножеством, но и для создания запросов специального вида. Так, возможность управлять числом записей, выводимых в ответ, часто используется совместно с возможностью упорядочивать записи.

Для управления числом записей, выводимых в ответ, можно воспользоваться кнопкой **«Набор значений»** (_____J) либо задать нужную величину для одноименного свойства в окне свойств запроса (рис. 2.66).

Рис. 2.66. Определение числа выводимых записей путем указания параметра «Набор значений» в окне «Свойства запроса».

Число записей, выводимых в ответ, можно задавать как абсолютным числом, так и в процентном отношении от общего числа отобранных записей.

На рис. 2.67 изображен запрос: «На какой кафедре самый маленький средний оклад сотрудников?». Для задания этого запроса сначала традиционным способом определим средний оклад на каждой кафедре (запрос с подгруппировкой). Затем упорядочим ответ по возрастанию полученного поля (средний оклад). Если выбрать одну первую запись из полученного множества, то и получится искомый ответ.

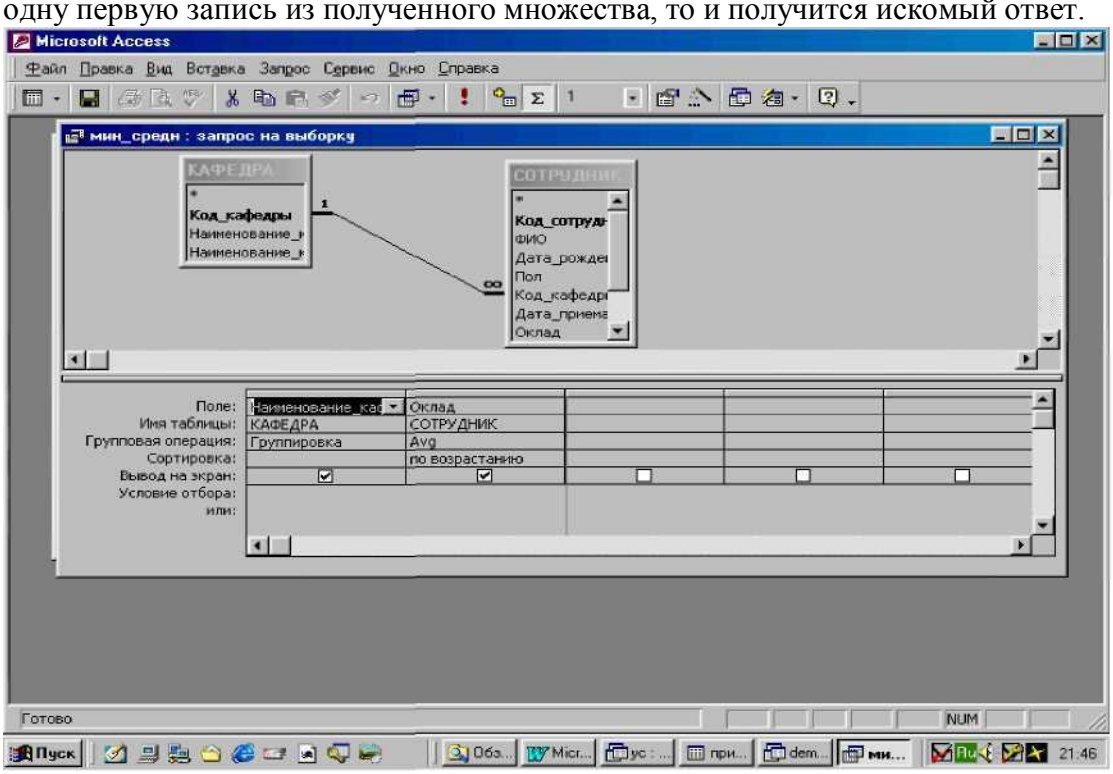


Рис. 2.67. Отбор определенного числа записей с предварительной сортировкой

2.3. Экранные формы

2.3.1. Создание простой многотабличной и многостраничной формы

Как мы видели в предыдущей главе, после описания таблицы можно сразу вводить в нее данные. Но такой способ имеет многие очевидные недостатки. Поэтому для этих целей обычно используются так называемые экранные формы.

Форму можно создавать несколькими разными способами. Для того чтобы создать новую форму, надо выбрать вкладку **«Формы»** в окне базы данных и нажать кнопку **«Создать».** После этого появиться окно **«Новая форма»** (рис. 2.68**).**

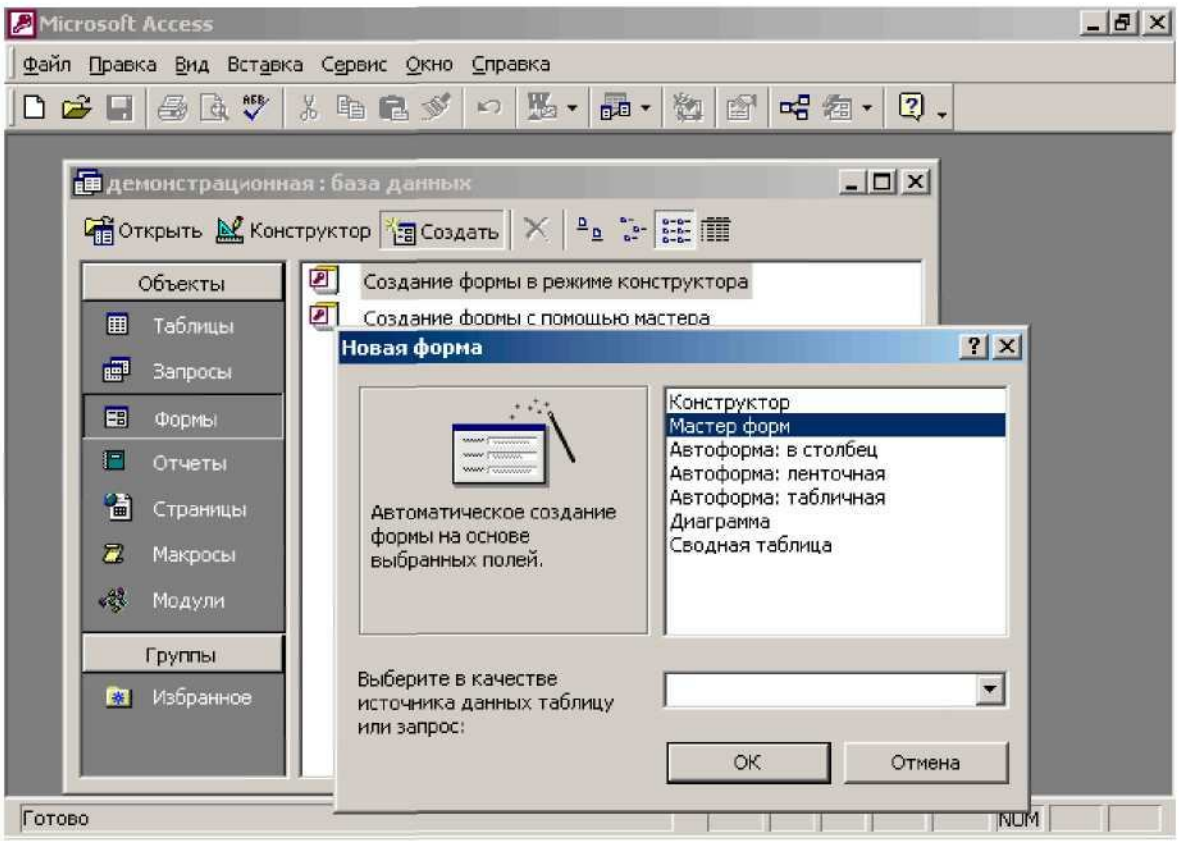


Рис. 2.68. Начальный экран создания форм

Создание простой формы с помощью мастера

При создании формы, связанной с таблицами базы данных, лучше сначала воспользоваться помощью **«Мастера»,** а потом модифицировать полученную таким образом форму. Это ускорит и упростит процесс создания форм.

В окне **«Новая форма»** кроме выбора способа создания/вида формы можно определить и источник данных для создаваемой формы (рис. 2.68). Из окна **«Новая форма»** можно выбрать только одну таблицу в качестве источника данных для формы. Мы в качестве примера создадим форму для таблицы «СОТРУДНИК». Выбор источника можно осуществить и на следующем шаге (рис. 2.69) в окошке **«Таблицы/запросы».**

Если источником должны являться несколько таблиц, то можно поступить несколькими разными способами, которые будут рассмотрены позднее.

Следующим шагом при создании форм является выбор полей, которые будут включены в экранную форму (рис. 2.69). Поля могут переноситься в форму по одному и все сразу. В первом случае надо позиционироваться на нужное поле и нажать кнопку с одинарной стрелкой, направленной вправо. Чтобы перенести все поля, надо воспользоваться кнопкой с двойной стрелкой. Когда поля включены в форму, на этапе определения состава полей их можно исключить, воспользовавшись кнопками, со стрелками, направленными влево.

При использовании *мастера* при создании формы выбор таблицы и хотя бы одного поля обязательны.

Источником данных для формы могут быть не только таблица/таблицы, но и запросы. Напомним, что в запросах можно использовать знак «звездочка» (*). При его использовании все поля исходной таблицы выводятся в ответ. Если уже после создания такого запроса в таблицу будут добавлены новые поля, то они попадут в ответ. Если же на основе такого запроса создана экранная форма, то в нее войдут те поля, которые на момент создания экранной формы содержались в таблице. Поля, которые вставлены в таблицу после создания экранной формы, в запрос, являющийся источником будут попадать, а в экранной форме отражаться не будут. Поэтому, если необходимо, чтобы все поля исходной таблицы были отображены в форме, то созданную форму придется корректировать «вручную».

Предположим, что мы хотим создать форму на основе таблицы «СОТРУДНИК» и включить в эту форму все поля исходной таблицы (рис. 2.69).

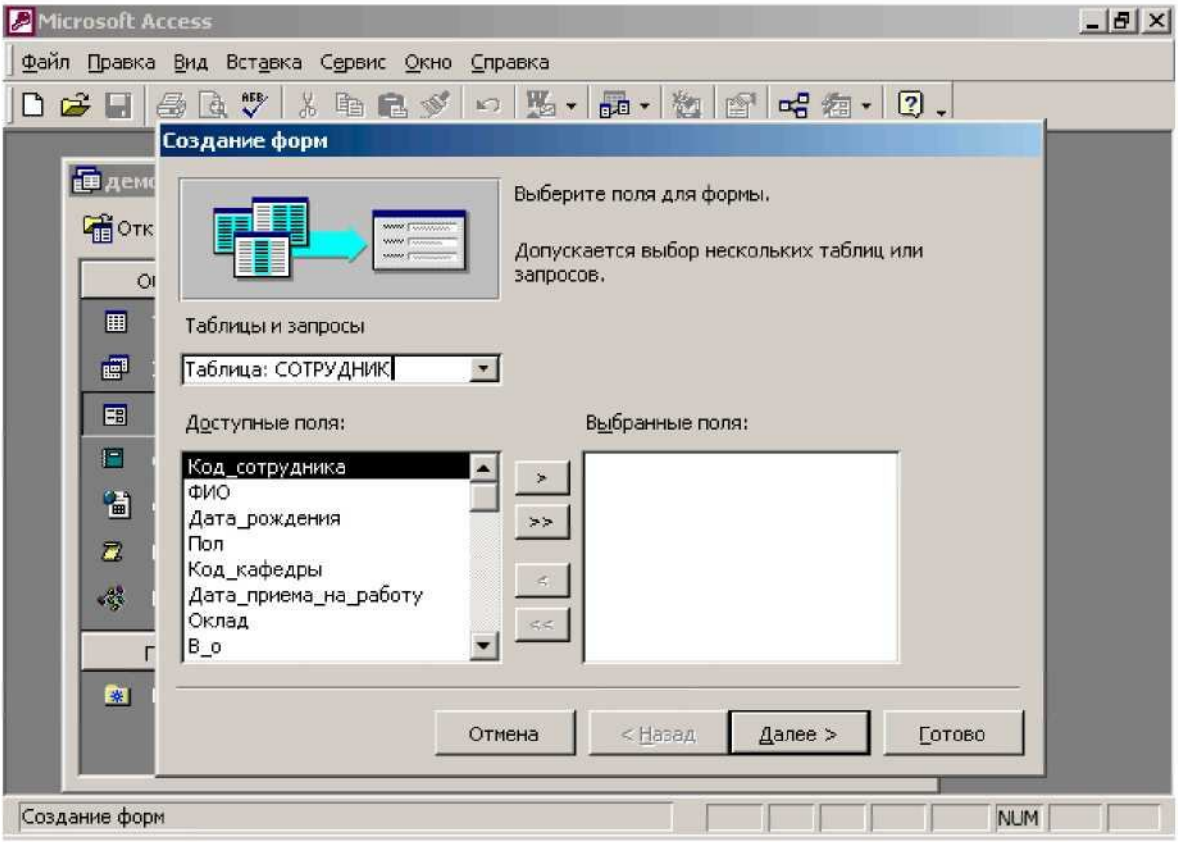


Рис. 2.69. Выбор полей, включаемых в форму

После определения состава полей, включенных в форму, следует выбрать внешний вид формы (рис. 2.70). Удобнее сначала выбрать вид **«в один столбец»5,** а затем разместить элементы формы по экрану так, как это удобно для последующей работы с формой.

Название вида формы «в один столбец» несколько условно. Действительно при данном выборе создает экран «анкетной» формы: один под другим располагаются пары «название поля/ содержание поля». Но когда полей много, и они все не умещаются в один столбец на экране, то система автоматически помещает их в два, три и т. п. столбца.

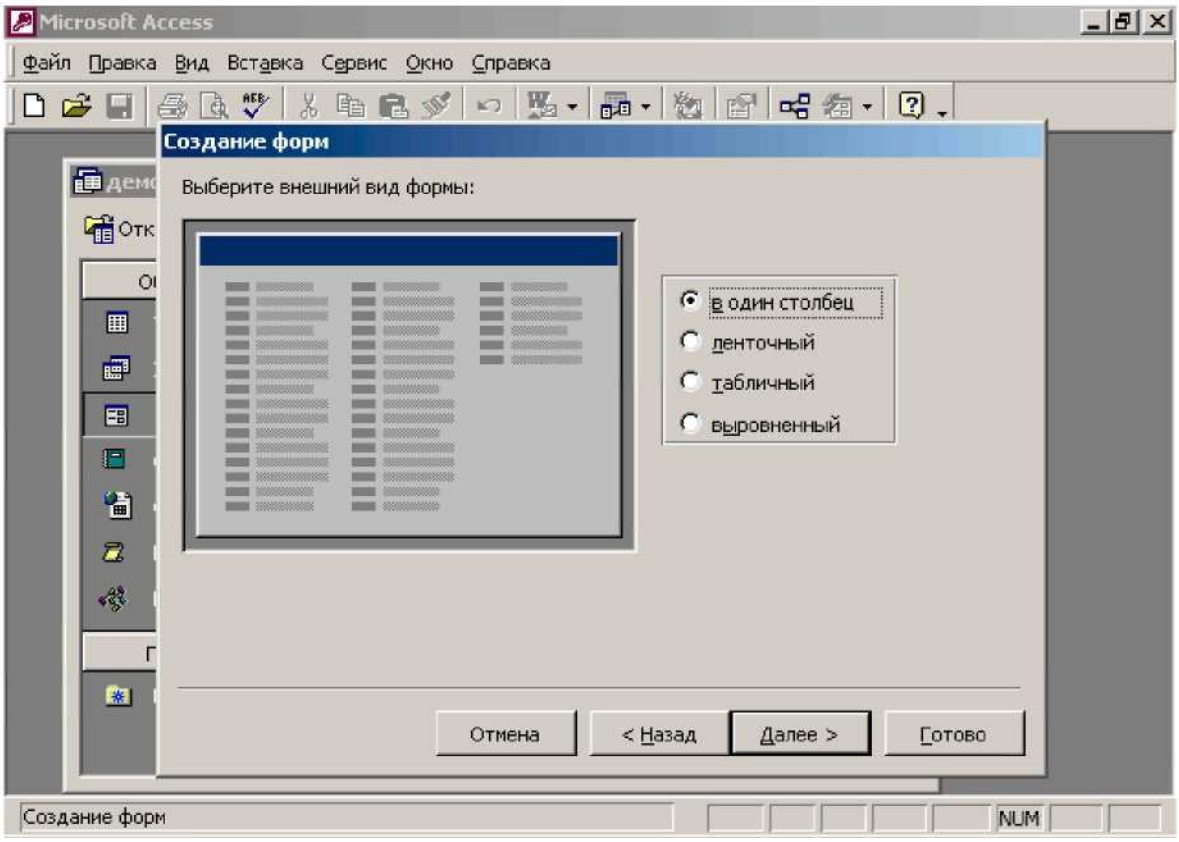


Рис. 2.70. Выбор внешнего вида формы

Следующим шагом является выбор стиля формы (рис. 3.71). Стиль экранной формы позволяет выбрать цвет и «фактуру» формы, т.е. позволяет решить чисто оформительские проблемы.

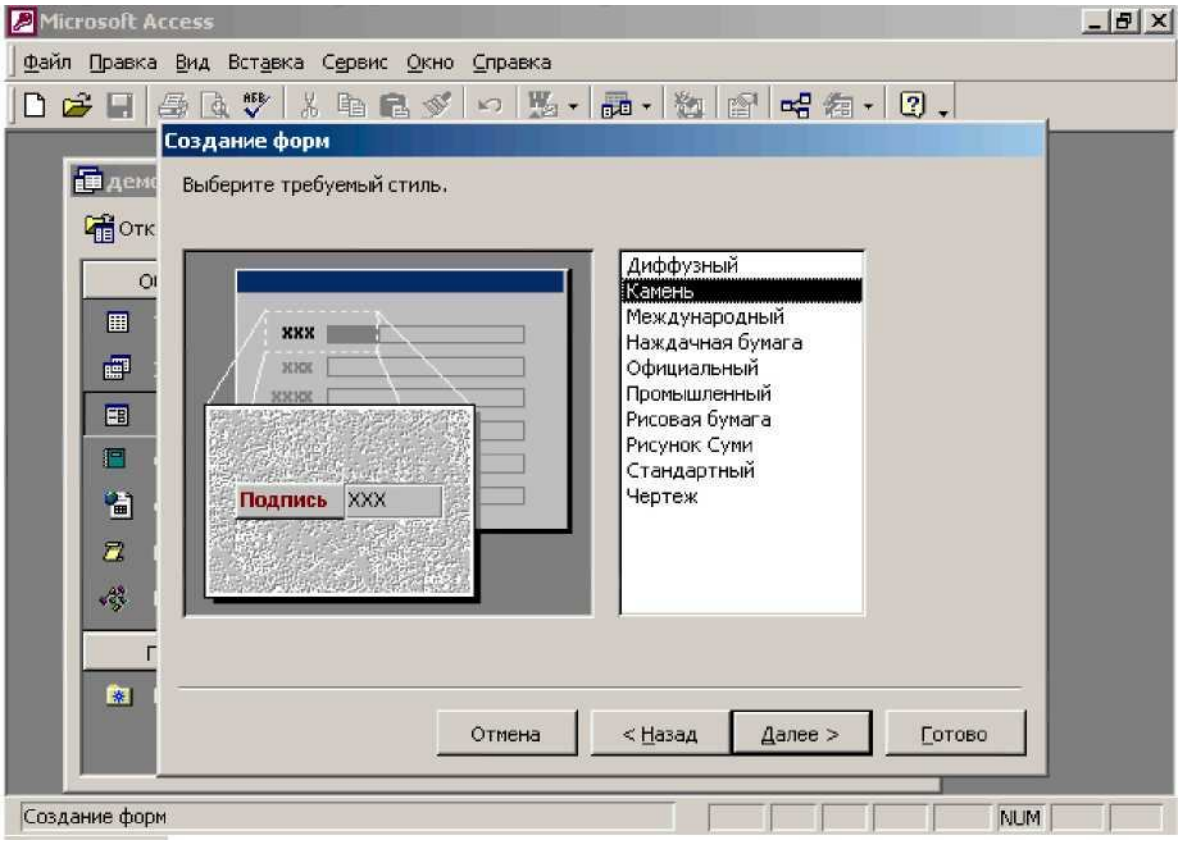


Рис. 2.71. Выбор стиля формы

Создание формы завершается заданием его имени (рис. 2.72). По умолчанию дается имя, совпадающее с именем источника данных.

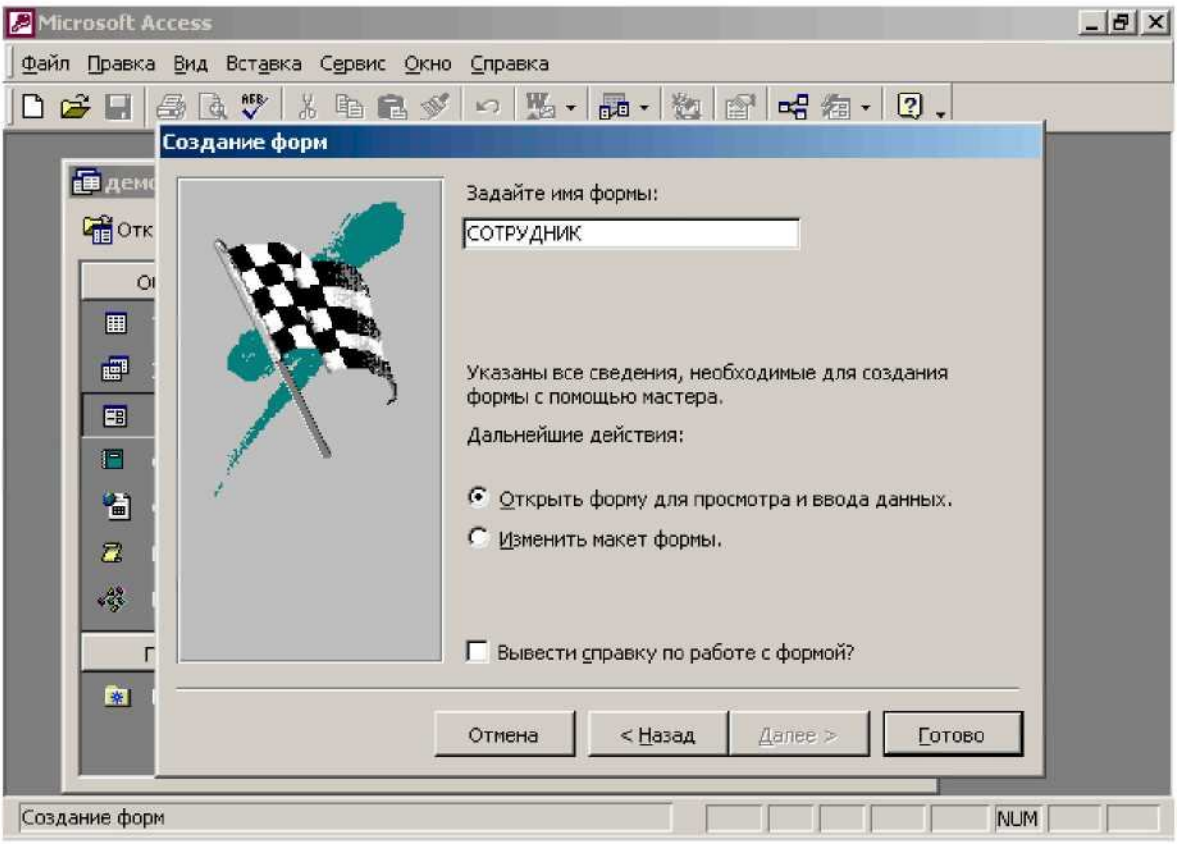


Рис. 2.72. Задание названия формы

Созданная экранная форма может не полностью соответствовать потребностям разработчика. Чтобы ее подправить можно сразу перейти в режим конструктора, выбрав альтернативу **«изменение макета формы»,** а можно это сделать и потом, перейдя в режим конструктора из режима формы, или открыв ранее созданную форму в режиме конструктора.

Создание многотабличной формы

Многотабличную форму также можно создать при помощи **«Мастера».** При создании форм, источником данных для которых являются несколько таблиц, можно находясь в окне «Создание форм/выбор полей» (рис. 2.69) последовательно выбирать таблицы, являющиеся источником данных, и поля из них. При этом таблицы должны быть обязательно предварительно связаны между собой.

Внимание!!! *Порядок, в котором выбираются таблицы при создании «многотабличной формы» имеет большое значение.*

Предположим, что мы хотим создать экранную форму, содержащую сведения из таблиц «СОТРУДНИК» и «ДЕТИ». Если мы выберем сначала таблицу «СОТРУДНИК» и из нее - поле «ФИО», а затем таблицу «ДЕТИ» и из нее все поля, то создастся составная форма, которую удобно использовать для ввода данных в таблицу «ДЕТИ». Создание многотабличной формы начинается также как и создание однотабличной формы. На первом шаге выбираем способ создания формы - **«Мастер форм»,** и таблицу-источник - «СОТРУДНИК». На втором шаге из таблицы «СОТРУДНИК» выберем поле «ФИО». После этого вернемся к окошку **«Таблицы и запросы»** и выберем в нем вторую, соподчиненную таблицу - «ДЕТИ» (рис. 2.73). Перенесем все поля из этой таблицы в форму.

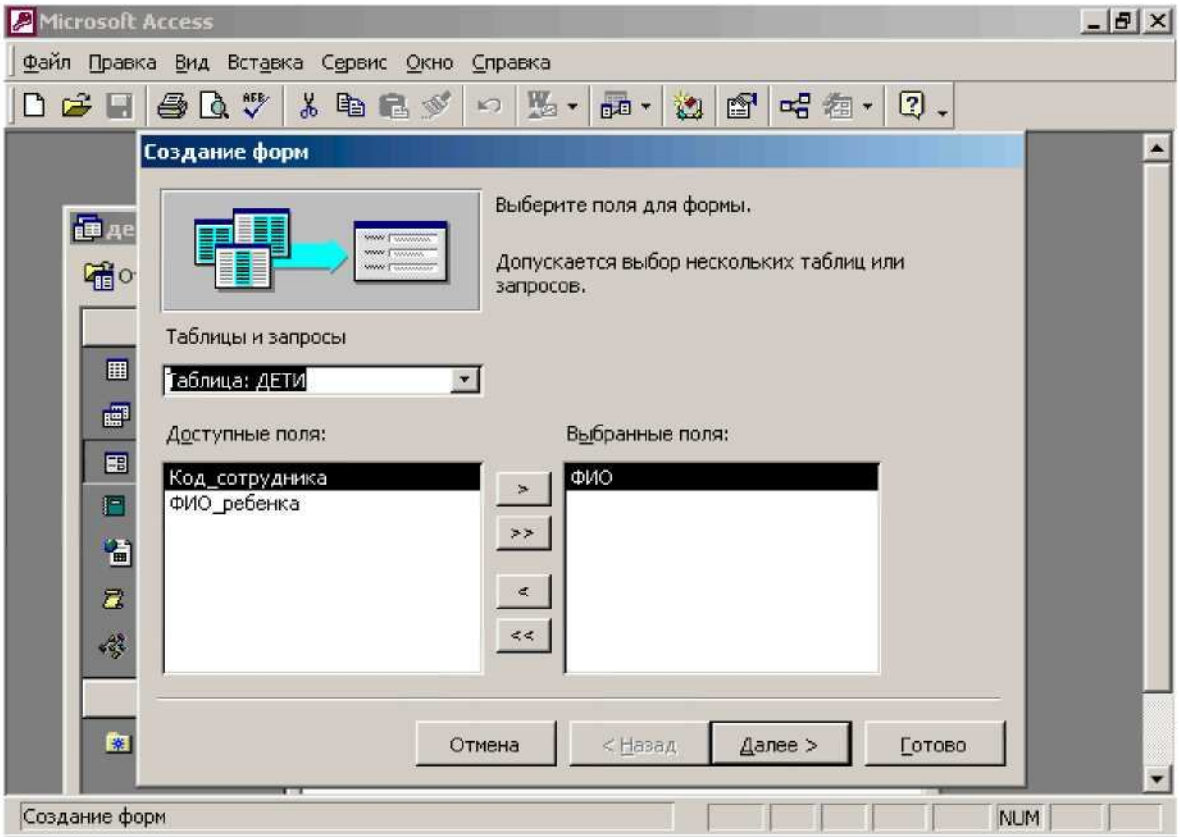


Рис.2.73. Создание многотабличной формы (выбор полей)

Дальнейшая последовательность шагов создания составной формы представлена на рис. 2.74-2.76.

Сначала выбирается вид представления данных - подчиненная или связанная форма (рис. 2.74). Для наших целей подходит подчиненная форма.

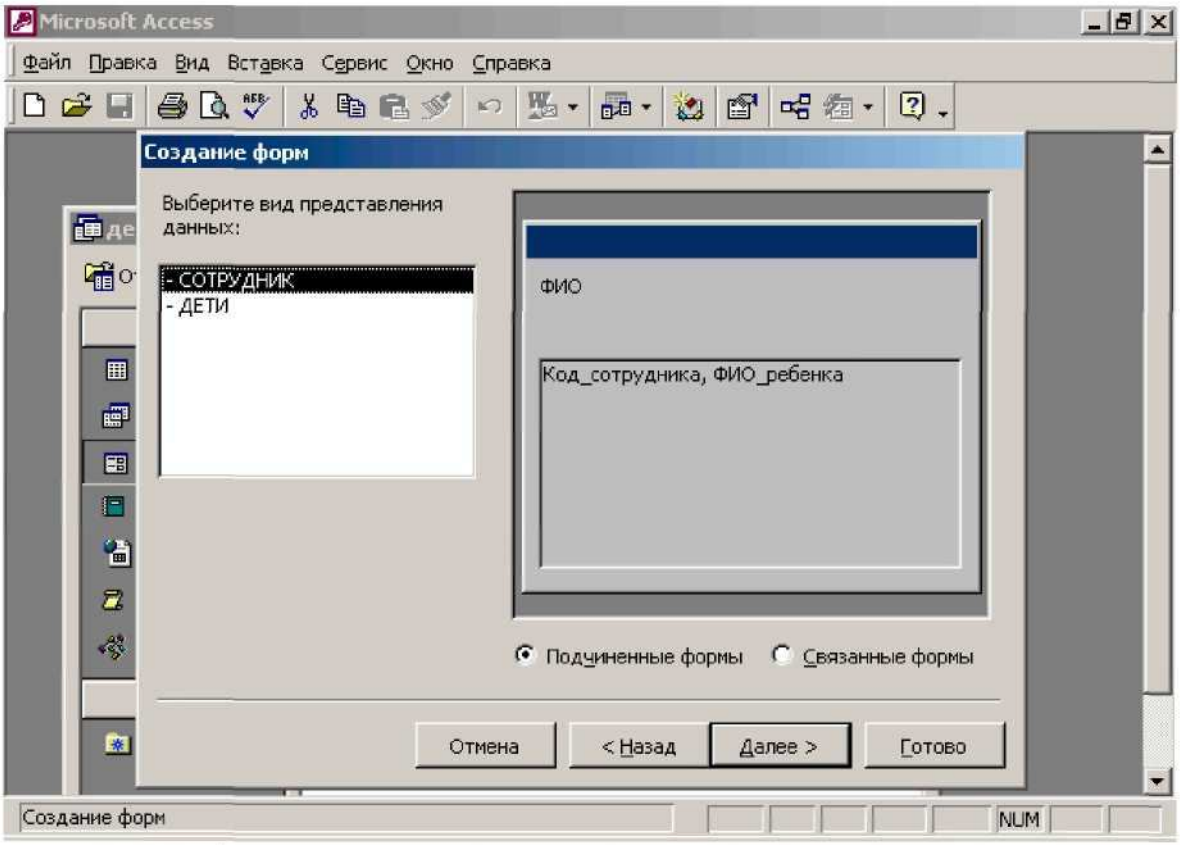


Рис. 2.74. Создание многотабличной формы (выбор вида представления)

Затем выбирается вид подчиненной формы (рис. 3.8).

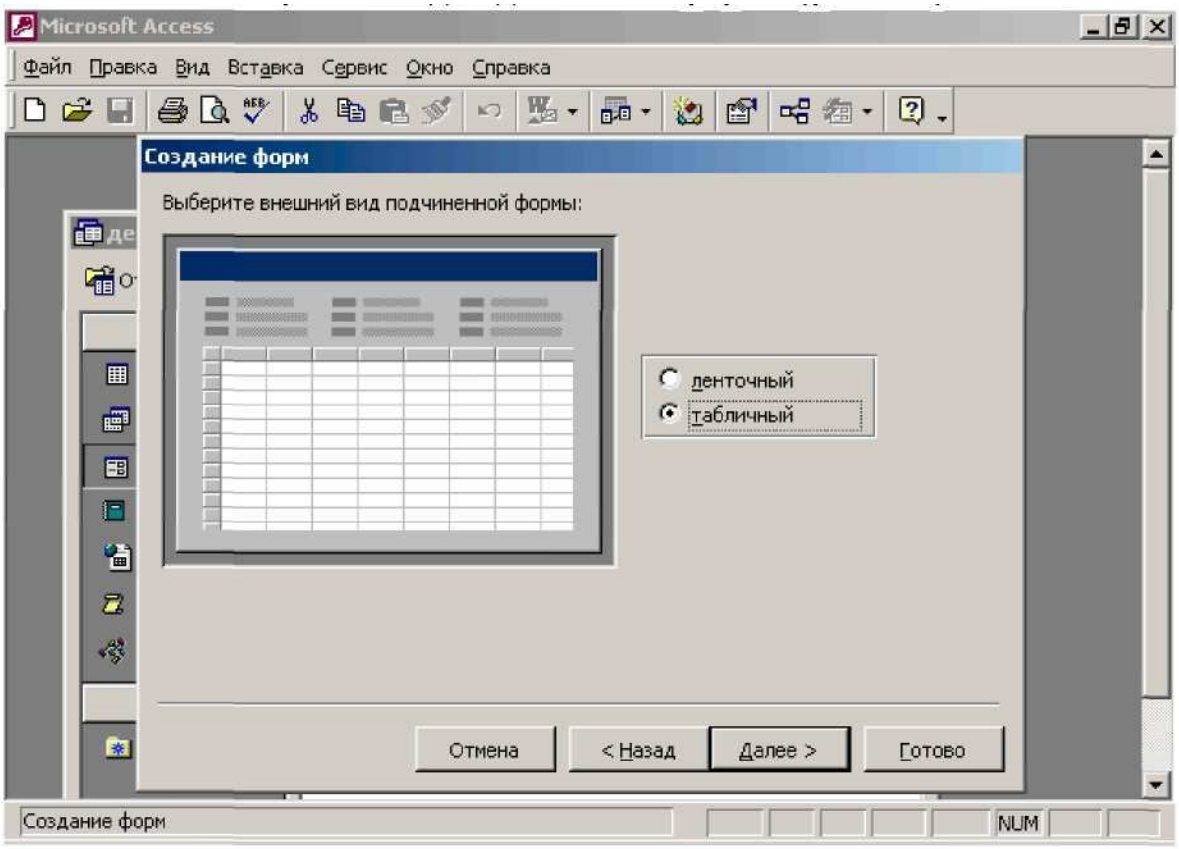


Рис.2.75. Создание многотабличной формы (выбор вида подчиненной формы)

Следующий шаг(выбор стиля) совпадает с аналогичным шагом при создании однотабличной формы.

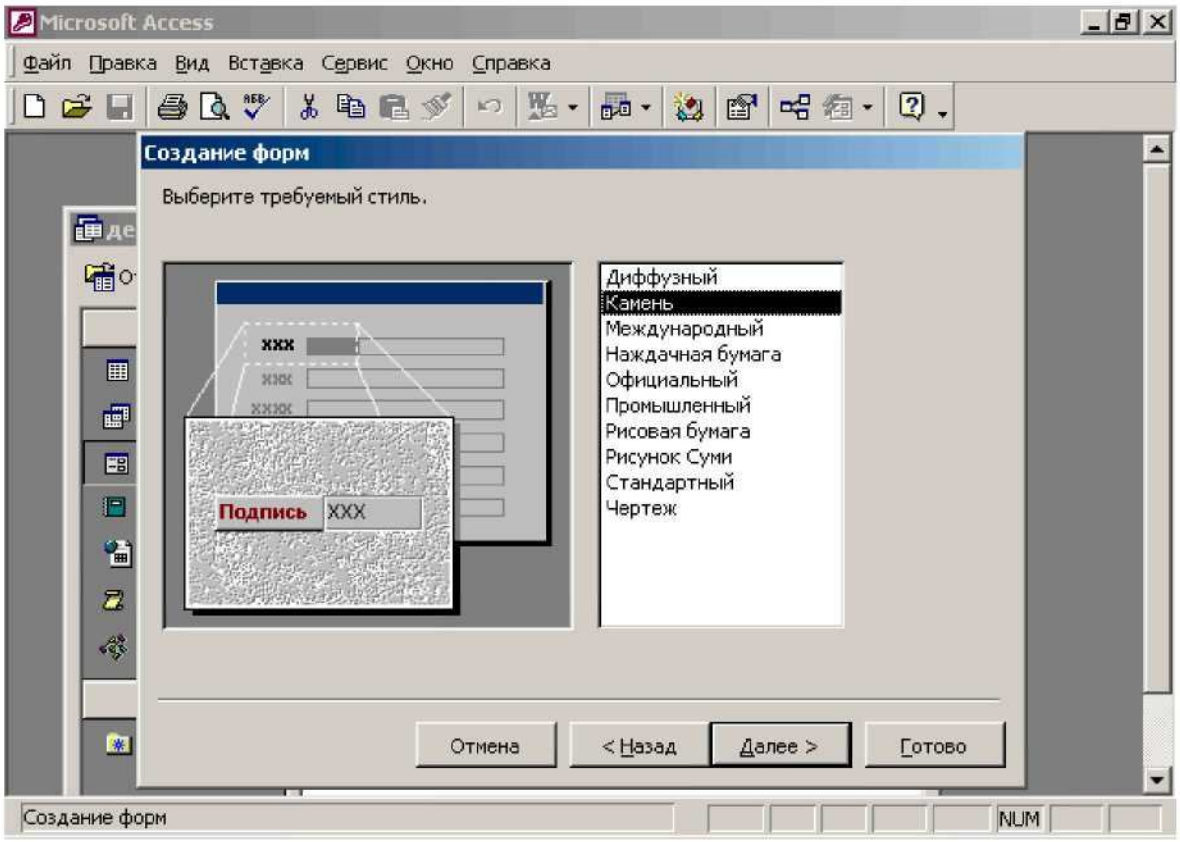


Рис. 2.76. Создание многотабличной формы (выбор стиля)

При создании многотабличной формы система создаст описание двух форм: основной и подчиненной. Поэтому при завершении создания формы надо задать соответственно имена для этих двух форм. На рис. 2.77 представлен вид составной формы в режиме формы.

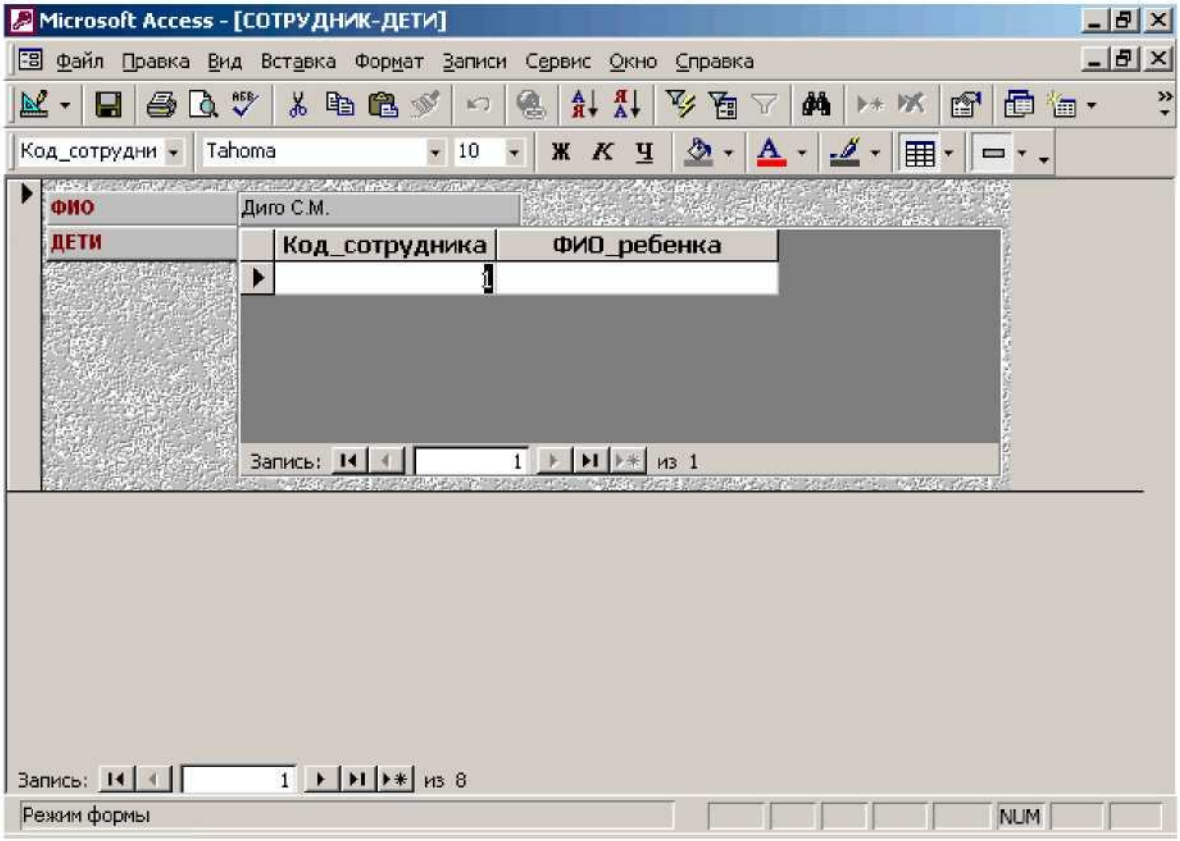


Рис. 2.77. Составная форма в режиме формы

При вводе данных в подчиненную форму код сотрудника вводится в подчиненную таблицу автоматически. Как видно, в подчиненной форме выводятся только записи, связанные с записью в главной форме.

При использовании составных экранных форм можно легко перемещаться как по записям подчиненной формы, так и по записям главной формы. Для этого используется соответствующий набор кнопок перехода. Кроме того, для быстрого позиционирования на нужную запись можно воспользоваться возможностью поиска в БД. Для этого можно выбрать позицию меню **«Правка/Найти»** (или воспользоваться соответствующей кнопкой инструментального меню) и в появившемся окне ввести условие поиска.

Если при построении многотабличной формы сначала выбрать таблицу, находящуюся на стороне «многие» в отношении «1:М» (в нашем примере это таблица «ДЕТИ»), а потом таблицу, находящуюся на стороне «1» (в нашем примере это таблица «СОТРУДНИК»), то многотабличная экранная форма также будет создана, но это будет совсем иная форма, чем та, что изображена на рис. 2.77. Это будет форма, в которой отображаются записи подчиненной таблицы (т. е. таблицы «ДЕТИ»), к которым присоединены поля из соответствующей записи основной таблицы. Никакой соподчиненности форм при этом не наблюдается. Создается только одно описание формы, не два, как в предыдущем случае.

Другим способом создания многотабличной формы является создание запроса, отбирающего те поля из связанных таблиц, которые будут помещаться в форму, и использование этого запроса в качестве источника для формы.

Создание форм, состоящих из нескольких страниц

Существует несколько причин, по которым бывает необходимо/целесообразно разнести данные, размещаемые в экранной форме, по нескольким страницам. Это может быть в случае, когда элементов в форме много, и размещение всех их на одном экране слишком загромождает его; либо может быть вызвано желанием сгруппировать поля и показывать каждую группу отдельно и др. Для достижения этих целей можно либо, воспользовавшись элементом управления **«Разрыв страницы»,** указать, в каких местах должен быть переход на следующую страницу, либо создать форму с несколькими закладками (рис. 2.78).

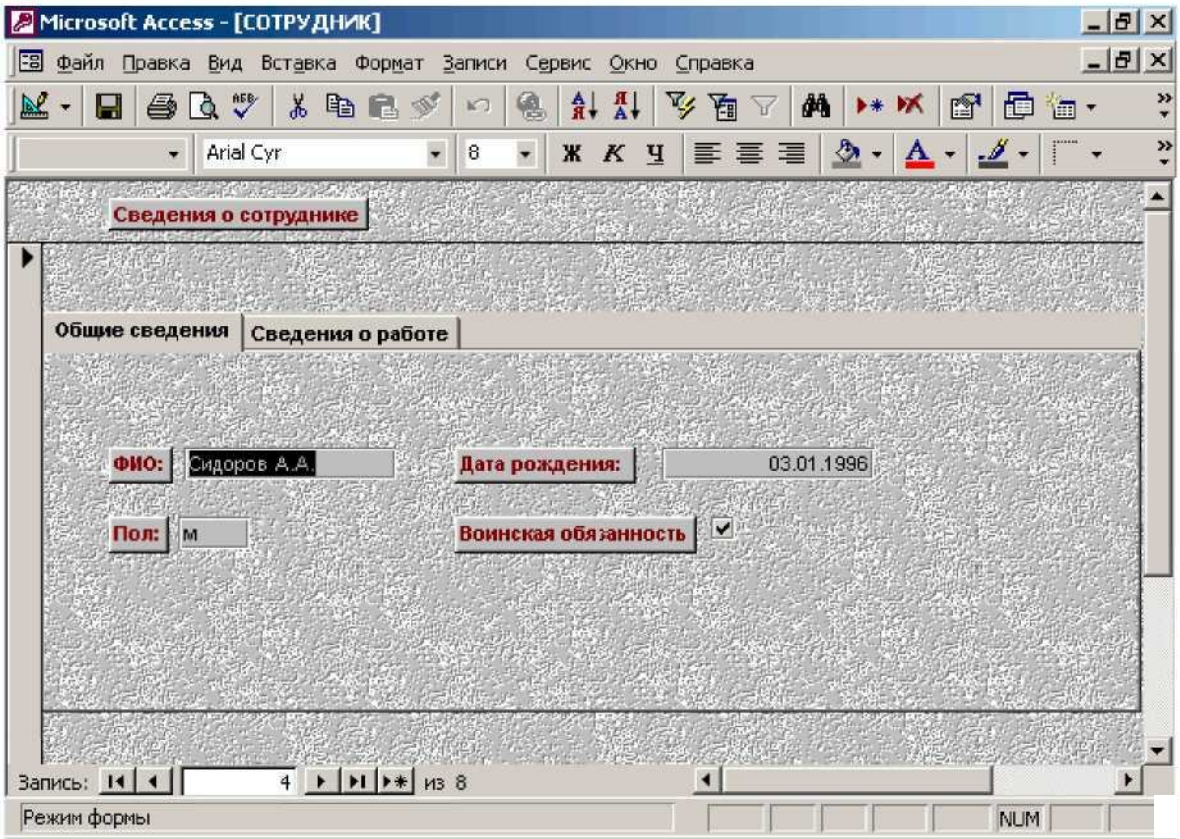


Рис.2.78.Экранная форма с закладками. Режим формы

Элемент управления **«Разрыв страницы»** используется для указания горизонтальных разрывов между элементами управления в форме. Для перехода к странице, находящейся над или под указанным разрывом, используются клавиши PAGE UP или PAGE DOWN.

Для того чтобы создать форму с несколькими закладками, можно воспользоваться элементом управления **«Набор вкладок»** ().

При выборе этой возможности в форме создаются две закладки. В каждую из них можно переместить те элементы, которые необходимо. Если необходимо создать большее число вкладок, то следует, находясь в зоне вкладок нажать правую клавишу мыши и в появившемся контекстном меню выбрать позицию **«Добавить вкладку»** (рис. 2.79). Для включения элементов, нужно выбрать необходимую вкладку и нажать

кнопку **«Список полей»** () на панели инструментов. Из появившегося ниспадающего списка надо выбрать имя нужного поля, позиционировавшись на нем, нажать левую клавишу мыши и не отпуская ее, переместить элемент в нужное место вкладки.

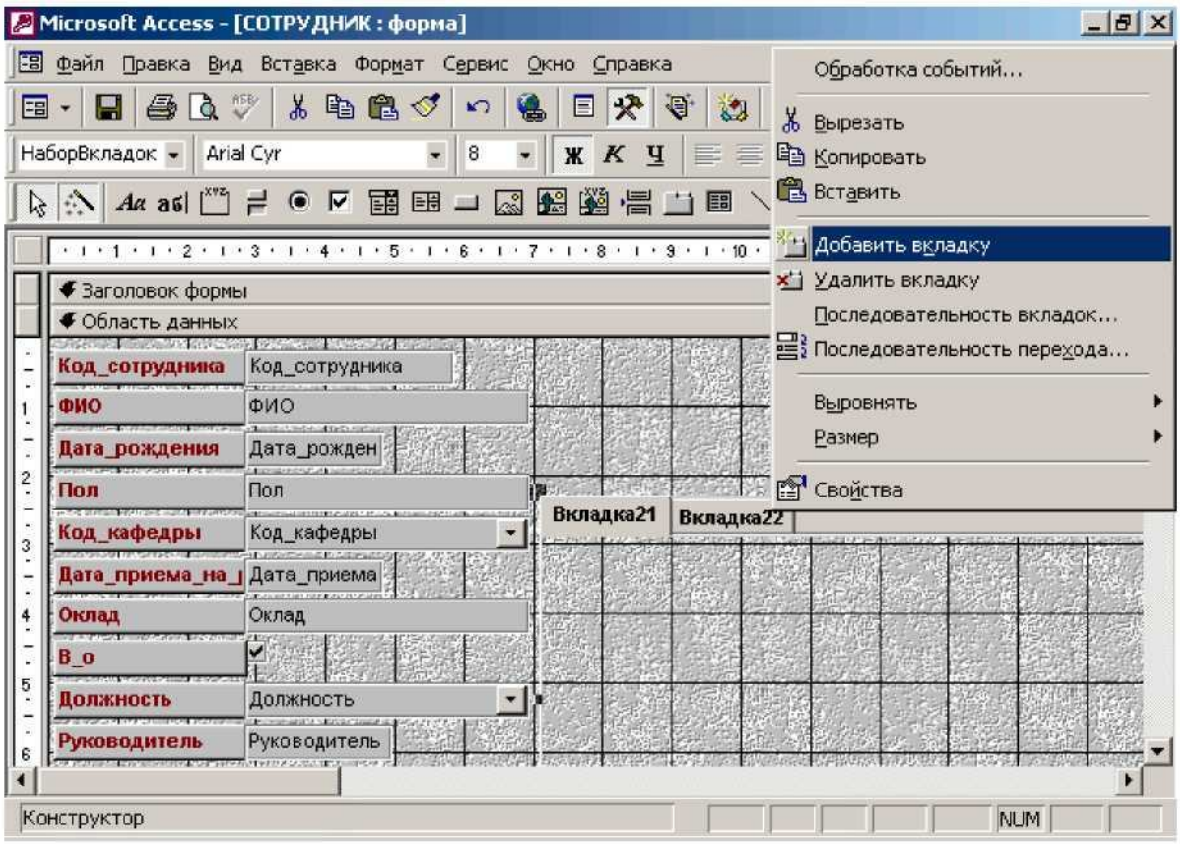


Рис. 2.79. Добавление вкладок

Для того чтобы *изменить название вкладки,* надо воспользоваться правой кнопкой мыши, в появившемся контекстном меню выбрать позицию **«Свойства»** и в свойстве **«Имя»** записать требуемое название (рис. 2.80).

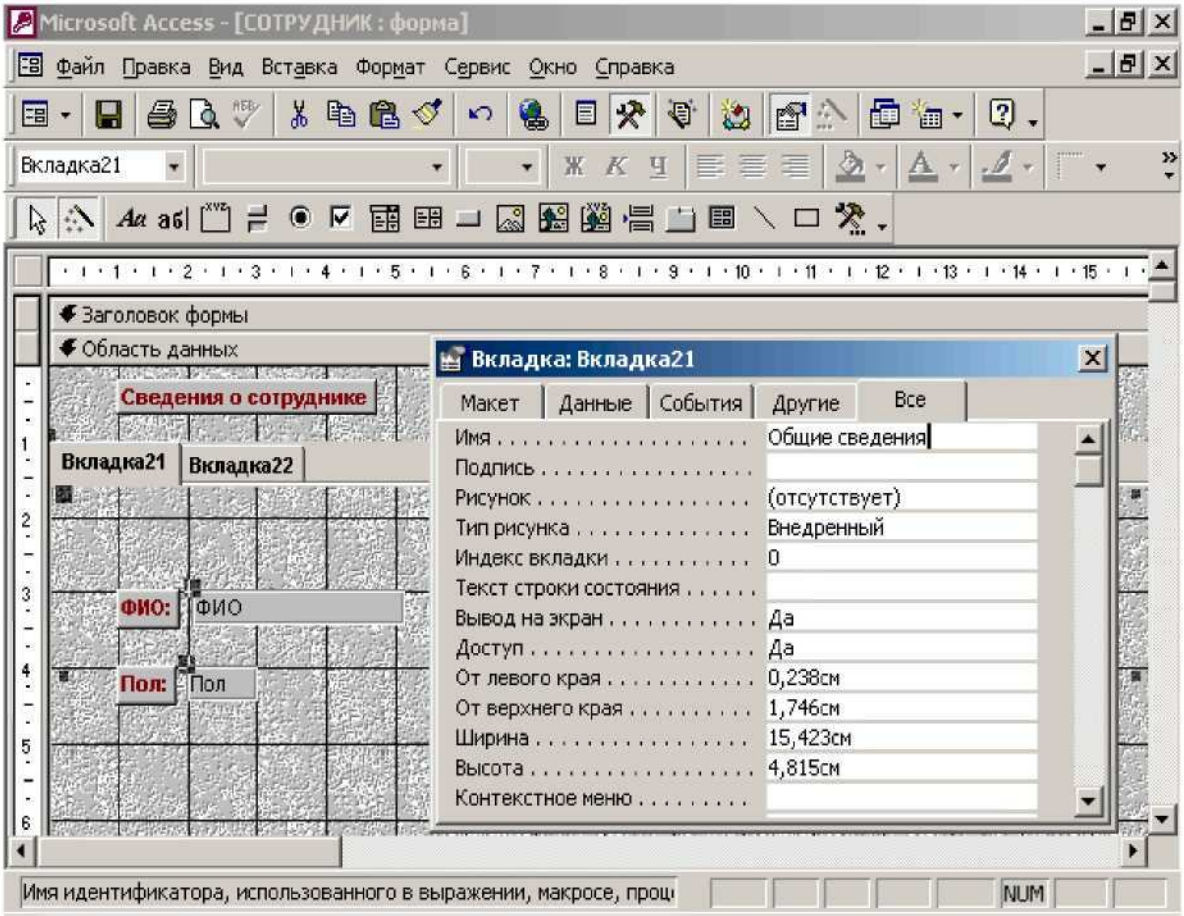


Рис. 2.80. Изменение названия вкладки

2.3.2. Корректировка формы в режиме «Конструктора»

Как отмечалось выше, форма, созданная мастером, может быть скорректирована. Для этого надо одним из указанных ранее способов перейти в режим конструктора. Экранная форма, полученная с помощью **«Мастера»,** в режиме конструктора будет иметь вид, представленный на рис. 2.81.

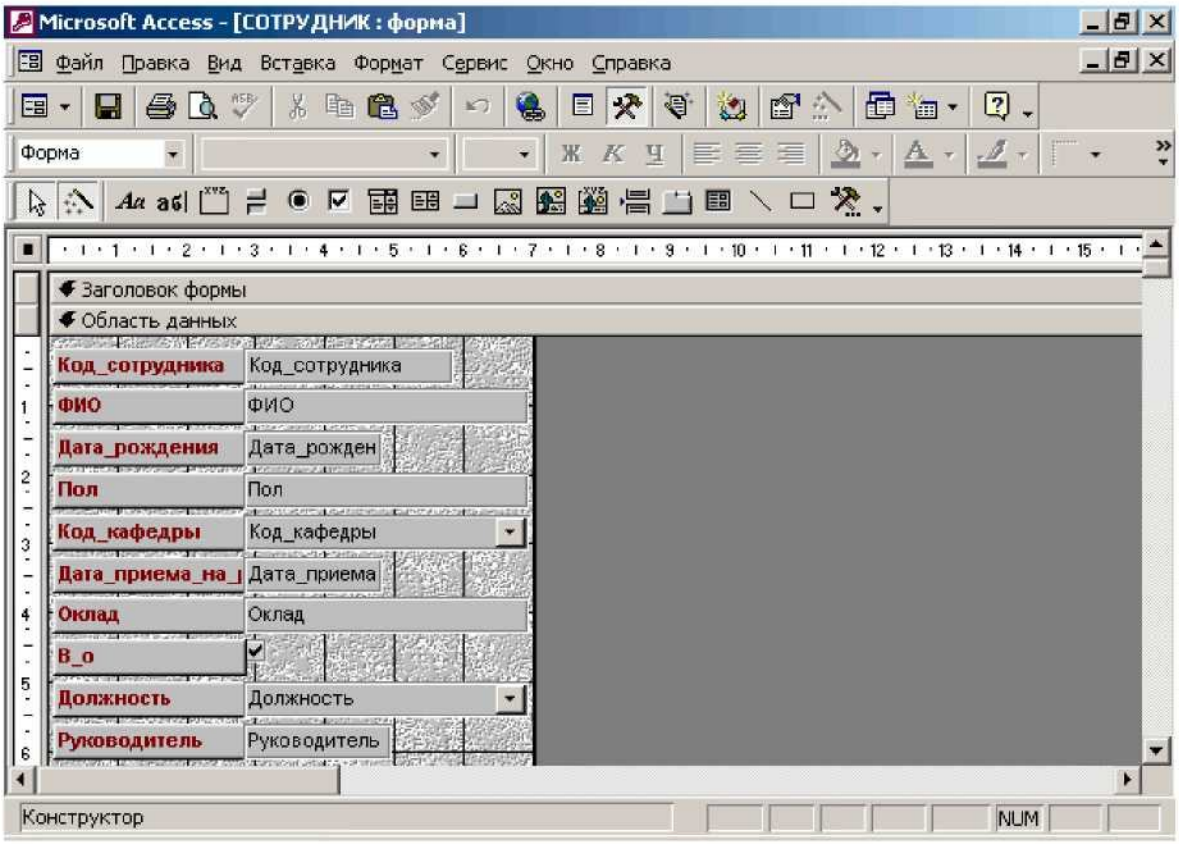


Рис. 2.81. Вид экранной формы в режиме конструктора

В верхней части экрана видны три инструментальных панели: форматирования, конструктора форм и панели элементов. Первая из панелей стандартная для всех офисных приложений MS. Многие из элементов второй панели также привычны и понятны. Некоторые из специфичных кнопок и их применение будут пояснены ниже.

Как мы видим на рис. 2.81, все элементы полученной формы размещены в области данных. Область заголовка формы закрыта, но ее можно раскрыть и ввести в нее заголовок формы и другие данные, относящиеся ко всей форме. Так как назначение и способы работы с облас-

тями форм и отчетов одинаковы, а их использование в отчетах актуально, то эти вопросы рассмотрены в разделе учебного пособия, посвященном отчетам.

Изменения, связанные с уже включенными в форму элементами управления

Изменения, вносимые в исходную форму, могут быть разнообразными. Прежде всего, существует возможность перемещения, изменения размеров и выравнивания уже включенных в форму элементов управления. Для этого элемент/элементы, которые надо изменить, должны быть выделены. Для выделения элемента управления надо установить на него указатель и нажать кнопку мыши. Чтобы выделить несколько элементов управления, следует нажать клавишу SHIFT и, не отпуская ее, выделить все нужные элементы. Если выделяемые элементы находятся рядом, и их не разделяют никакие элементы, которые не должны входить в выделяемую группу, то можно нажать левую клавишу мыши и, не отпуская ее, охватить появившимся контуром все те элементы, которые надо выделить. Выделенные элементы имеют характерные маркеры по углам элемента и по серединам его сторон.

Перемещение

Чтобы переместить выделенный элемент/элементы, надо позиционироваться на них мышью и добиться, чтобы указатель приобрел форму ладони. И, держа нажатой левую клавишу мыши, переместить элементы на требуемое место. В этом случае перемещаться будут все выделенные элементы. Если из пары «подпись»-«элемент управления» надо переместить что-то одно, то надо позиционироваться на левый верхний угол нужного элемента и добиться, чтобы указатель приобрел форму «указательного пальца»: в этом случае будет передвигаться только этот элемент.

Изменение размера

Чтобы изменить размер элемента надо позиционироваться на границе элемента таким образом, чтобы указатель принял форму двунаправленной стрелки. Выбор направления стрелок (вверх-вниз, вправо-влево, по диагонали) зависит от того, как вы хотите изменить размер элемента.

Выравнивание

Чтобы выровнять выделенные элементы управления, в меню **«Формат»** надо выбрать команду **«Выровнять»,** а затем в появившемся списке выбрать способ выравнивания.

Удаление

Чтобы удалить выделенные элементы управления, надо нажать на клавишу ***«Del».***

Включение новых элементов в форму

Если первоначально были включены не все поля из таблицы (или вы нечаянно удалили нужный элемент), то нетрудно добавить в форму поля из таблицы, являющейся источником данных формы. Для этого надо нажать кнопку **«Список полей»** ( ) на панели инструментов. Из появившегося ниспадающего списка надо выбрать имя нужного поля, позиционировавшись на нем, нажать левую клавишу мыши и не отпуская ее, переместить элемент в нужное место формы.

Более сложным является вариант, когда для этих целей используется кнопка **«Поле»** (—) на панели элементов, а потом у вставленного в результате этого действия свободного элемента меняется соответствующим образом свойство **«Данные»,** а у его надписи - свойство **«Подпись».** Но такой способ лучше использовать только в том случае, когда иной путь невозможен, например, при выводе в форму вычисляемого поля. При создании вычисляемого поля в свойство **«Данные»** надо ввести выражение для вычисления значения этого поля.

Вычисляемые поля могут вводиться не только в те формы, которые используются для вывода информации, но и в те, которые используются для ввода данных в базу данных. Например, при вводе данных в таблицу «СОТРУДНИК», можно на экран вывести вычисляемое поле «ВОЗРАСТ»: это поле не будет храниться в таблице (в нее будет вводиться только «ДАТА РОЖДЕНИЯ»), а на экран автоматически при вводе даты рождения будет выводиться возраст, что удобно, например, для контроля вводимых данных.

Чтобы *ввести в форму текст,* надо нажать кнопку **«Надпись»** (___) на панели элементов и, не отпуская кнопку мыши, переместиться в то место в форме, куда следует поместить текст. После чего ввести нужный текст и нажать клавишу ENTER.

Кроме текста и полей в форму могут быть включены линии, квадраты, рисунки.

Изменение типа элемента управления

В экранной форме могут использоваться разные элементы управления, в том числе список, поле со списком, которые широко используются при создании экранных форм.

Если при создании таблицы поля были созданы как поля подстановки, то в форме, полученной в результате использования Мастера, этим полям будут соответствовать поля со списком.

Если поля были созданы как обычно, а вы хотите в форме использовать, например, поле со списком, то можно поступить для достижения желаемого результата разными способами.

Во-первых, можно изменить тип элемента управления в форме. Для этого в режиме конструктора надо выделить соответствующий элемент формы, щелкнуть правой клавишей мыши, и появившемся контекстном меню выбрать позицию **«Преобразовать элемент в…»,** как показано на рис.2.82. В табл. 2.1 приведены возможные варианты преобразования. Доступные преобразования будут зависеть от того, какой тип имеет выбранный элемент.

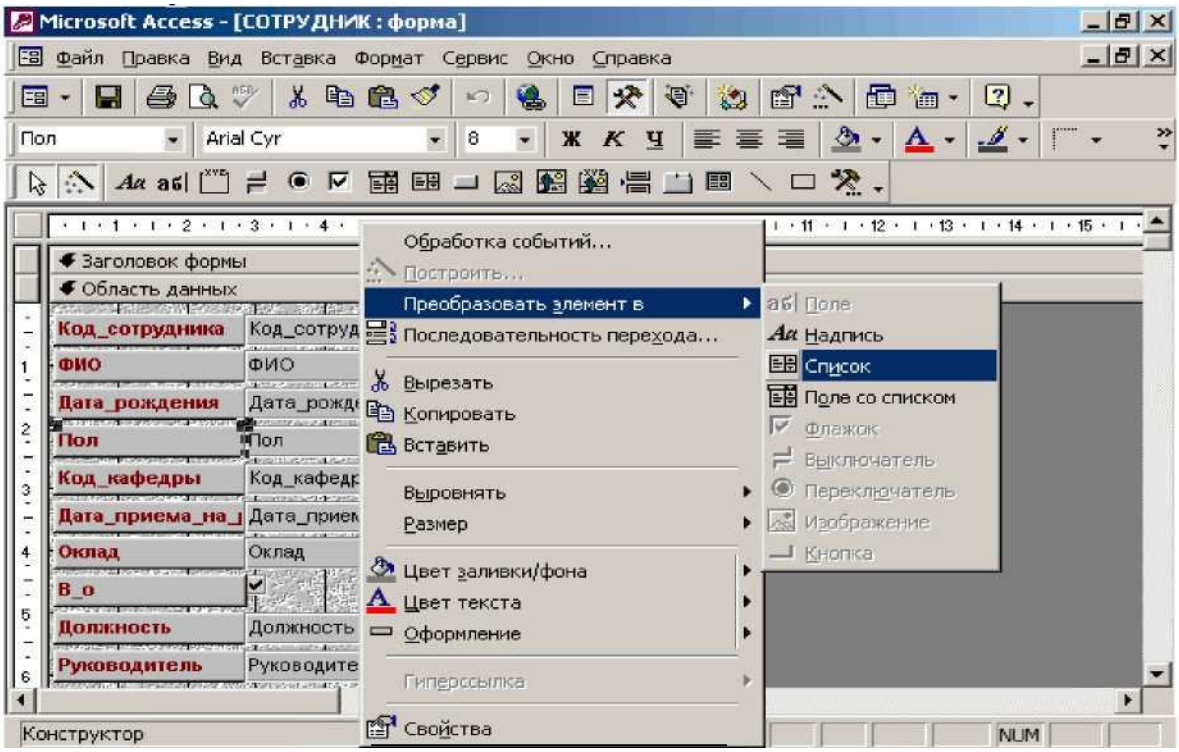


Рис. 2.82. Преобразование элемента

Таблица 2.5.

Возможные варианты преобразования типов элементов управления

	поле	надпись	список	поле со списком	флажок	выключатель	переключатель	Изображение	кнопка
поле (не логическое)		+	+	+					
надпись	+								
список	+			+					
поле со списком	+		+						
поле (логическое)					+	+	+		
- флажок						+	+		
- выключатель					+		+		
- переключатель					+	+			
изображение									
кнопка									

Но в некоторых из вариантов преобразования необходимо выполнить дополнительные шаги для того, чтобы достичь желаемого результата. Так, например, если обычное поле преобразовать в «список» или «поле со списком», то

автоматически список значений или связь с полем подстановки не появится. Поэтому надо соответствующим образом изменить свойства элемента. Для этого нужно позиционироваться на нужный элемент и нажать кнопку "Свойства" (^). Например, если мы хотим сделать элемент «Пол» списком с фиксированным набором значений «м» и «ж», то тип источника строк надо выбрать «Список значений», а в качестве источника строк через точку с запятой указать «м» и «ж» (рис. 2.83).

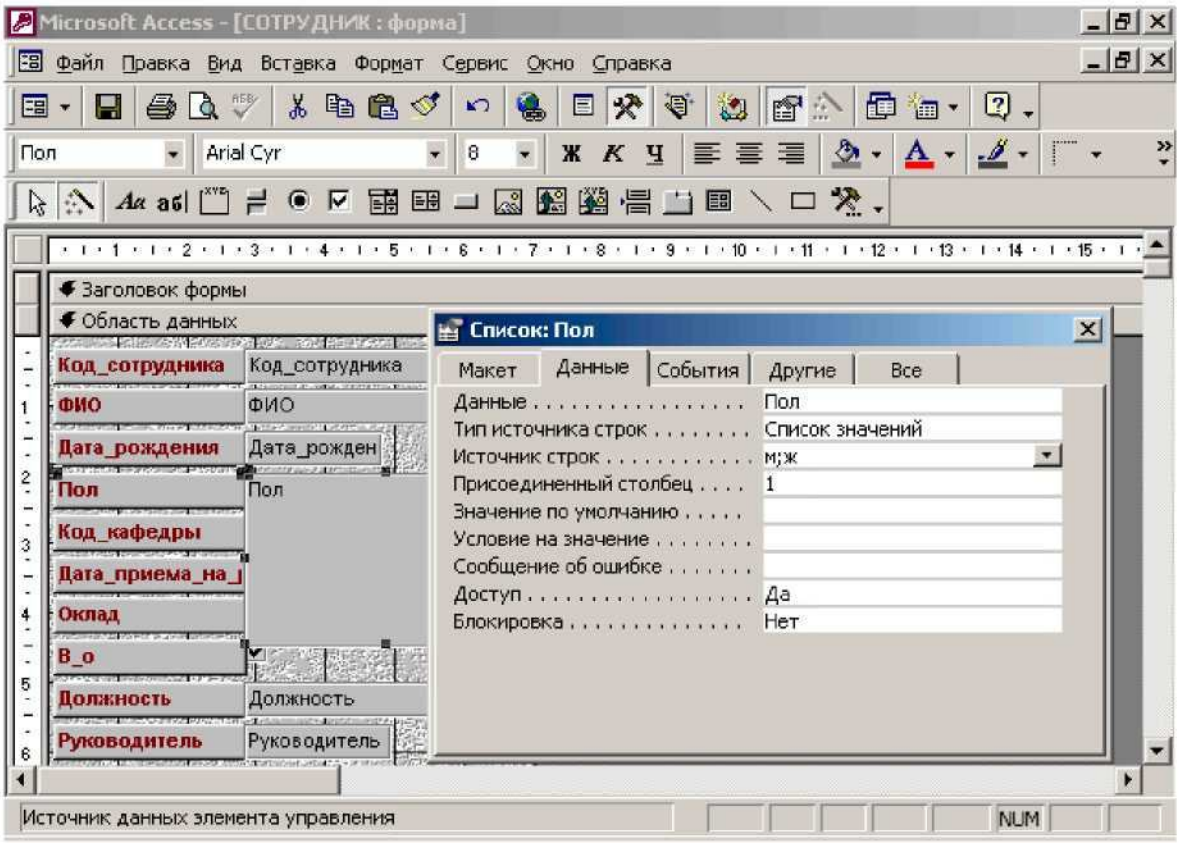


Рис. 2.83. Изменение свойств элемента при преобразовании типа элемента в «список»

Если элемент типа «поле» преобразуется в тип «поле со списком», то изменение свойств будет еще сложнее. Так, например, если мы хотим элемент, соответствующий полю «Код_кафедры» преобразовать в поле со списком, то вид «источника строк» будет «таблица/запрос»; в качестве источника строк следует выбрать таблицу «КАФЕДРА», после чего щелкнуть мышью на строке **«Источник строк»** и потом нажать кнопку с многоточием, чтобы вызвать построитель запросов (рис. 2.84).

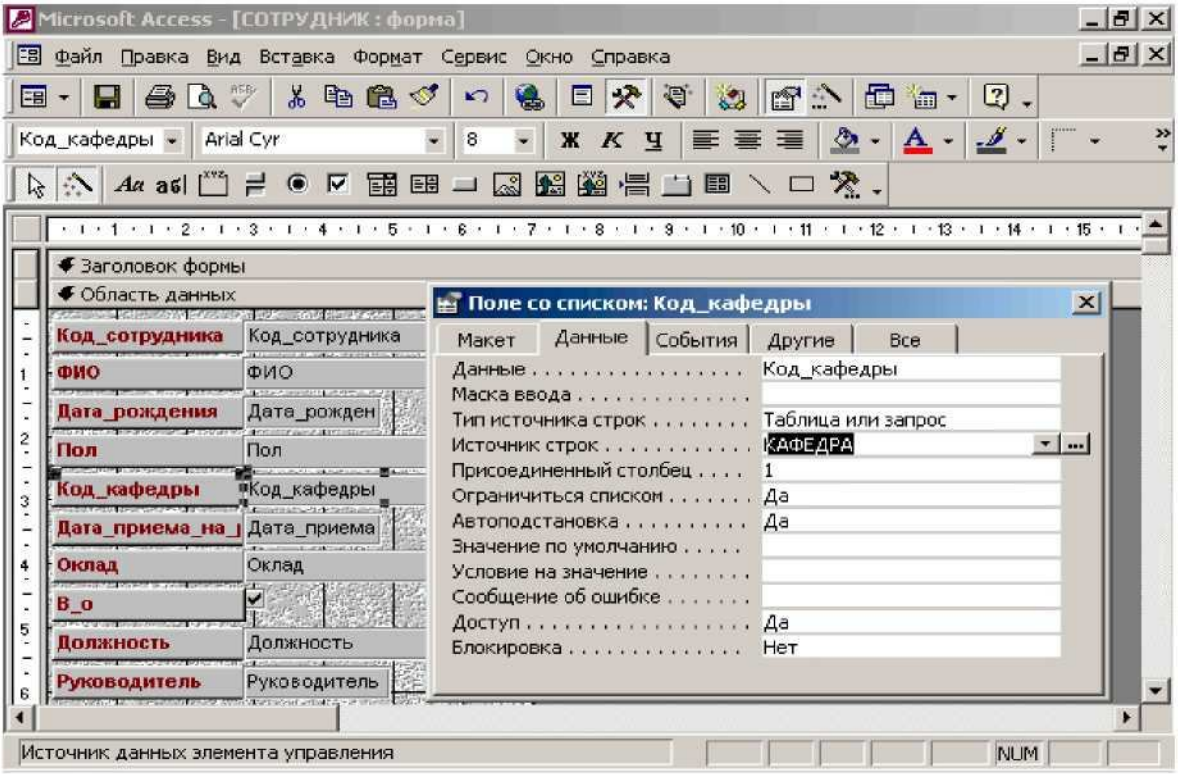


Рис. 2.84. Изменение свойств элемента при преобразовании типа элемента в «поле со списком»

Запрос, который вы будете строить, будет зависеть от того, только столбец подстановки будет выводиться на экран, или еще и поясняющий его столбец, т.е. в нашем примере столбец «КОД_КАФЕДРЫ» и «НАИМЕНОВАНИЕ_КАФЕДРЫ_ПОЛНОЕ». В этом случае запрос будет иметь вид, представленный на рис. 2.85.

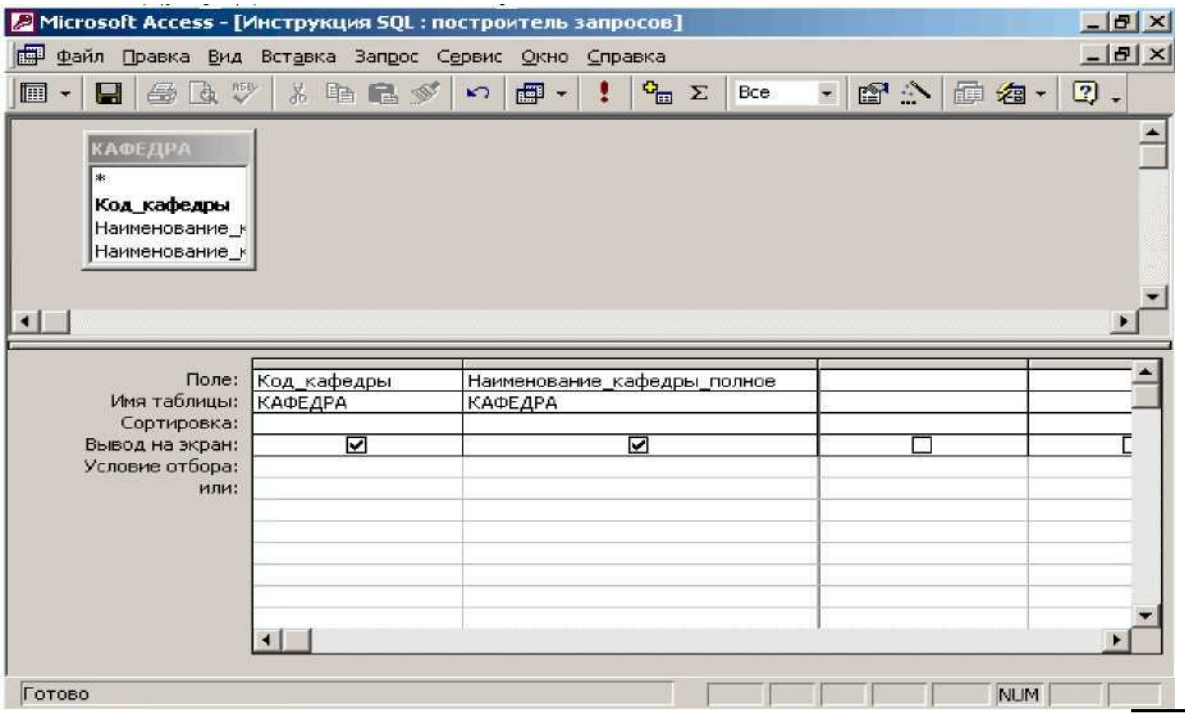


Рис. 2.85. Изменение свойств элемента при преобразовании типа элемента в «поле со списком» (продолжение)

Кроме того, может потребоваться изменение свойств «число столбцов» и «ширина столбцов».

Как мы видим, при отсутствии навыков такое преобразование является не совсем тривиальным.

Можно воспользоваться и другим вариантом «преобразования» типа элемента, а именно, удалить элемент из формы и создать его заново, выбрав на панели элементов элемент нужного типа.

Если вы работаете с версией Access, позволяющей при создании таблицы определять поле подстановки, то лучше воспользоваться этой возможностью.

Последовательность обхода полей

Последовательность обхода полей при работе с формой может отличаться от их расположения на экране. Для установления последовательности обхода полей можно, позиционировавшись на заголовок окна формы в режиме конструктора, нажать на правую клавишу мыши. При этом появиться всплывающее окно (рис. 2.86), в котором следует выбрать позицию **«Последовательность перехода…».**

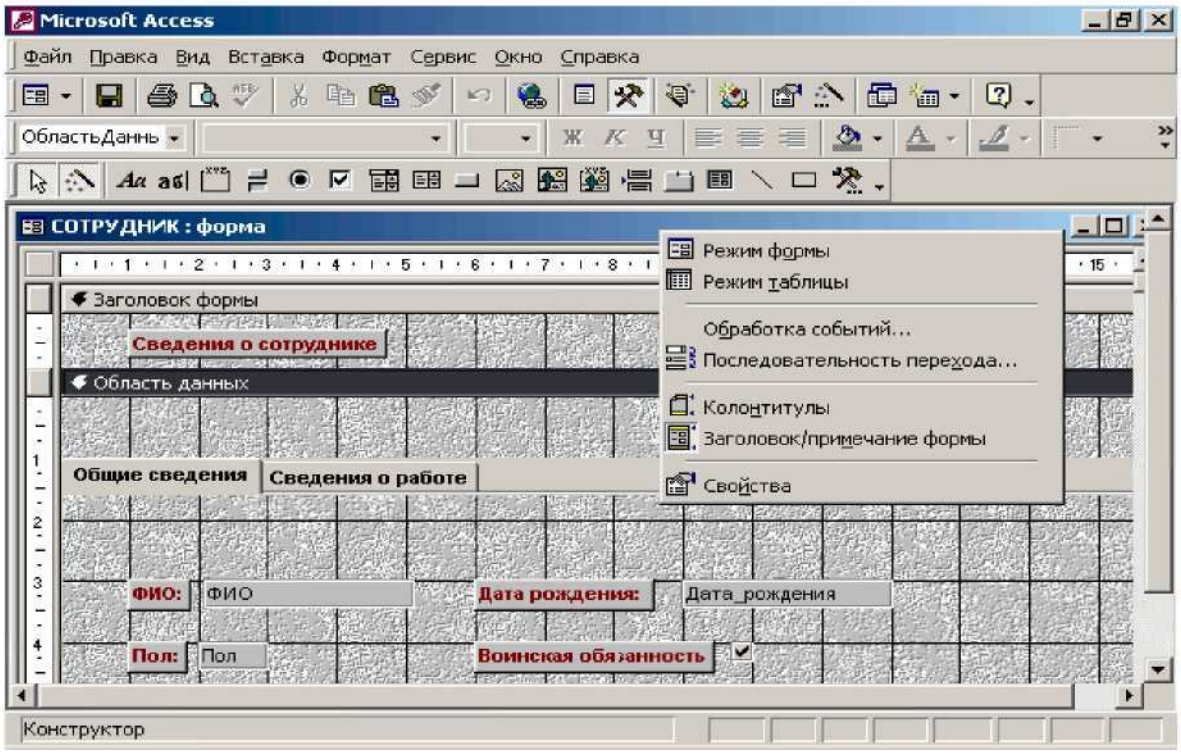


Рис. 2.86. Задание последовательности обхода полей (экран 1)

После чего на экране появиться окно (рис. 2.87), в котором перечислены поля, включенные в форму (если форма содержит несколько закладок, то будут выводиться только те элементы, которые включены в «активную» закладку).

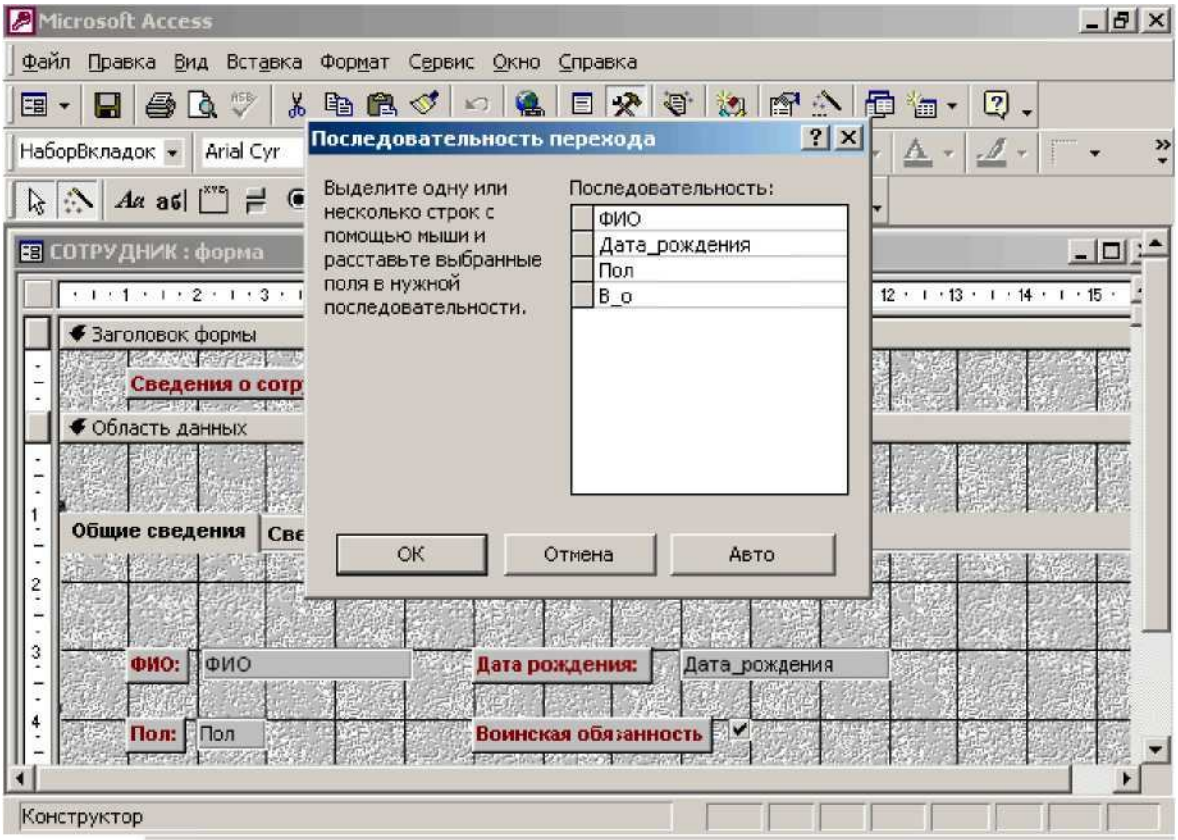


Рис.2.87. Задание последовательности обхода полей (экран 2*)*

2.3.3. Свойства формы

Кроме свойств, относящихся к каждому отдельному элементу формы, имеются свойства, относящиеся ко всей форме. Их можно с успехом использовать для создания дополнительных удобств при работе с формой, для обеспечения целостности базы данных и других целей. Для того чтобы посмотреть/скорректировать свойства формы, надо открыть форму в режиме конструктора и двойным нажатием кнопки мыши на области выделения формы открыть окно свойств формы. Перечень свойств формы обширен. Рассмотрим некоторые из них. Так, на вкладке **«Данные»** (рис. 2.88) имеется свойство **«Ввод данных».** Если выбрать для него значение «Да», то можно создать форму, использующуюся только для ввода данных. В этом случае в форме будет высвечиваться одна пустая запись, в которую можно вводить новые данные. Если вы хотите, чтобы выводились все записи, то значение этого свойства должно быть **«Нет».**

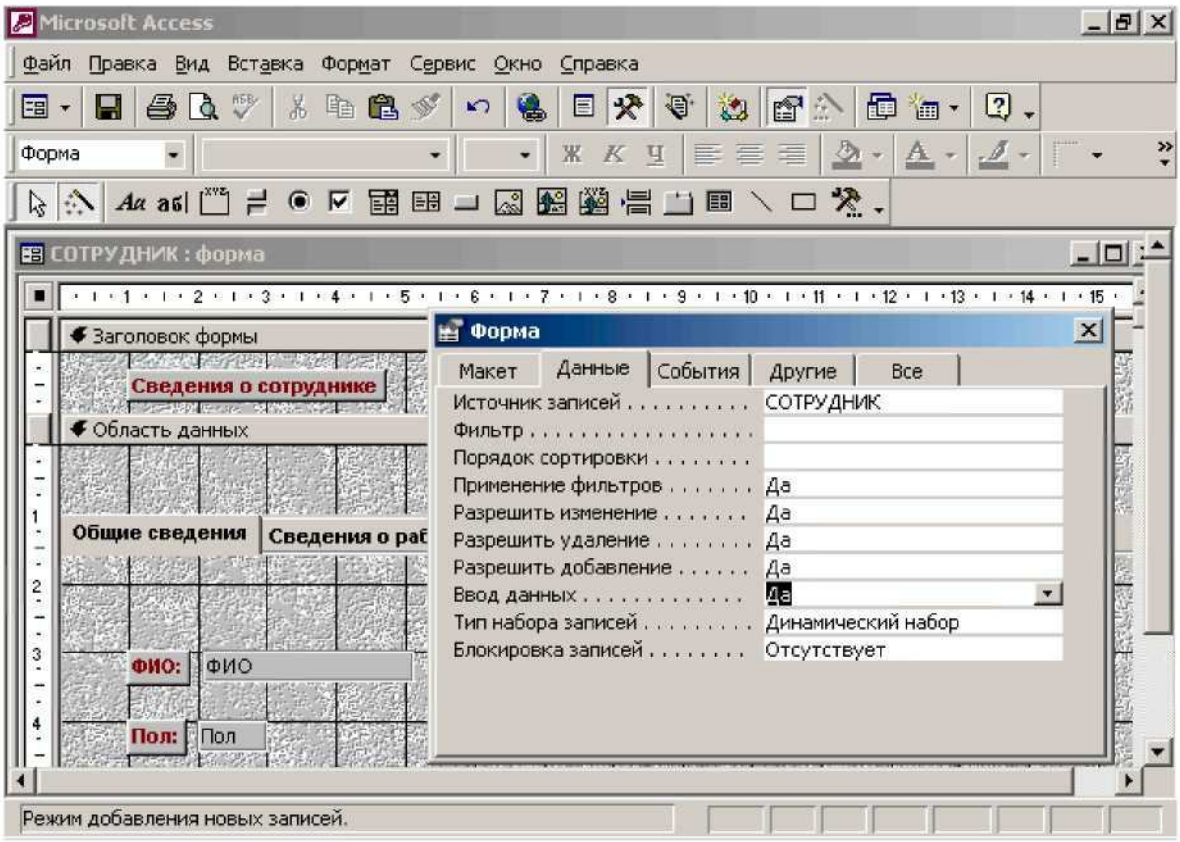


Рис.2. 88. Свойства формы

Можно, напротив, создать форму только для просмотра, запретив все корректировки.

2.4. Отчеты в Access

2.4.1. Способы создания отчетов

Создание отчетов является важной функцией, предоставляемой СУБД, так как именно отчеты позволяют представить данные из баз данных в удобном виде.

Для создания нового отчета в окне базы данных следует перейти на закладку **«Отчеты»,** нажать на кнопку **«Создать»** и в верхней части появившегося окна (рис. 2.89) выбрать способ создания отчета, а в нижней - указать таблицу или запрос, данные из которого будут выводиться в отчете. После чего следует нажать на кнопку **«ОК».**

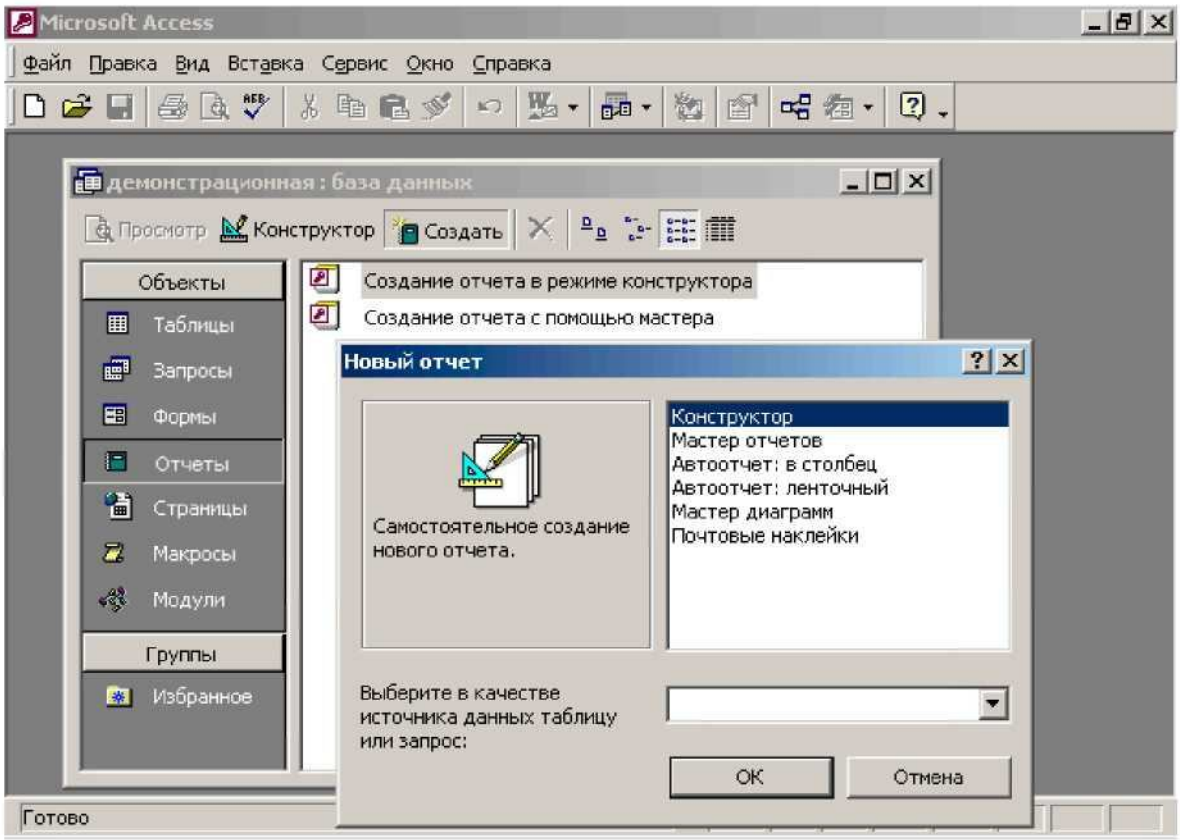


Рис. 2.89. Начальное окно создания нового отчета.

Создание отчета с помощью мастера

Также как и при создании формы, отчеты лучше создавать, пользуясь одним из мастеров, а потом, в случае необходимости, произвести желаемую корректировку формы отчета. Мастерами в данном случае можно считать все возможности, перечисленные в ниспадающем меню, изображенном на рис. 4.1, кроме позиции **«Конструктор».**

Простейшим способом создания отчетов является использование **«Автоотчетов».** При использовании этой возможности в отчет выводятся все поля выбранного источника данных (таблицы/запроса), названием отчета и его

заголовком становится название источника данных. Никакое вмешательство пользователя в процесс создания документа не предусматривается.

Более гибкой возможностью является использование «Мастера отчетов». Именно этот инструмент наиболее часто используется при первоначальном создании документа.

При любом способе создания отчета необходимо выбрать источник данных, на основе которого он будет формироваться. Источником может быть одна или несколько таблиц или запрос (рис. 2.90)

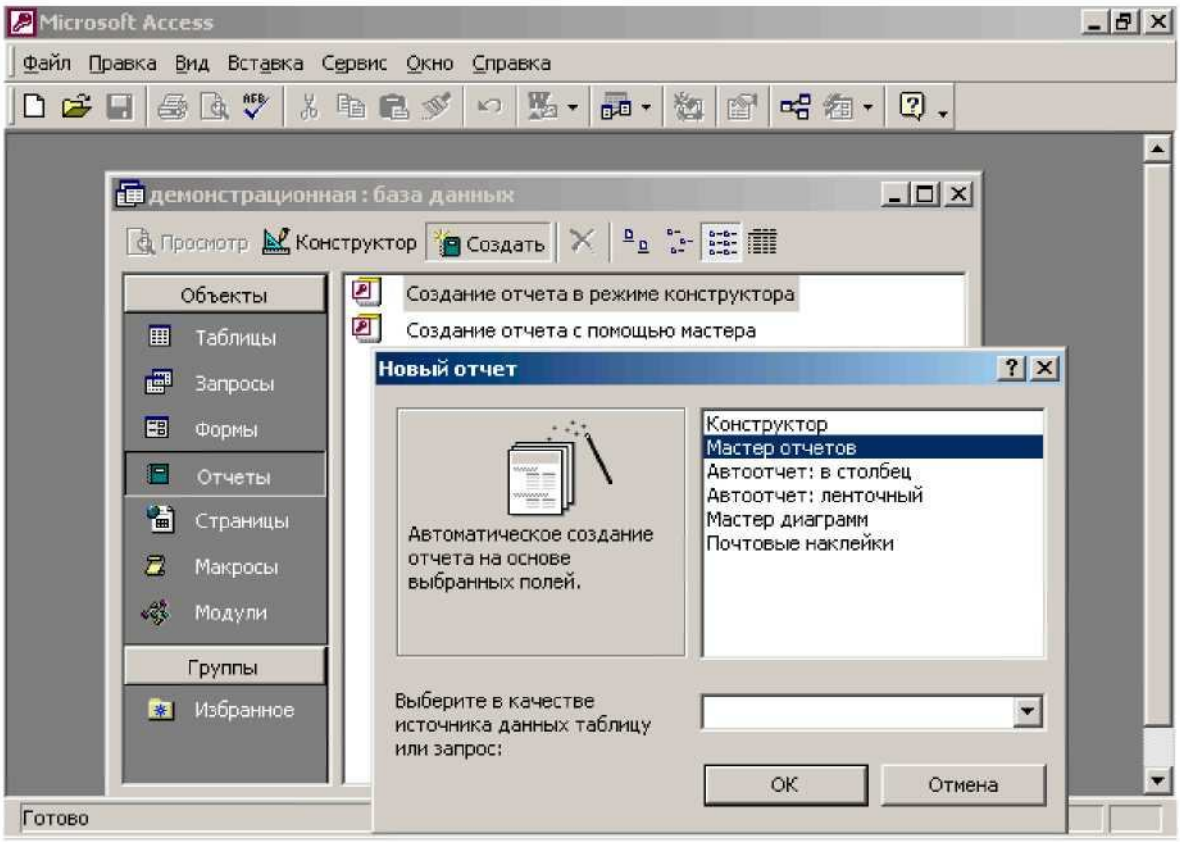


Рис. 2.90. Проектирование отчетов. Создание нового отчета. Выбор способа создания и источника

Если мы выберем «Мастер отчетов», то далее система предложит определить поля (рис. 2.91), которые будут входить в отчет. При выборе полей, входящих в отчет, можно использовать либо кнопку с двумя стрелками (в случае, если в отчет будут входить все или большинство полей), либо переносить поля по одному, используя кнопку с одной стрелкой. При этом следует *обратить внимание* на то, что поля следует переносить не в том порядке, в котором они располагаются в структуре исходной таблицы, а в том, в котором они будут использоваться в отчете. И хотя потом, в режиме конструктора можно менять положение тех или иных элементов в отчете, лучше сразу продумывать эти вопросы.

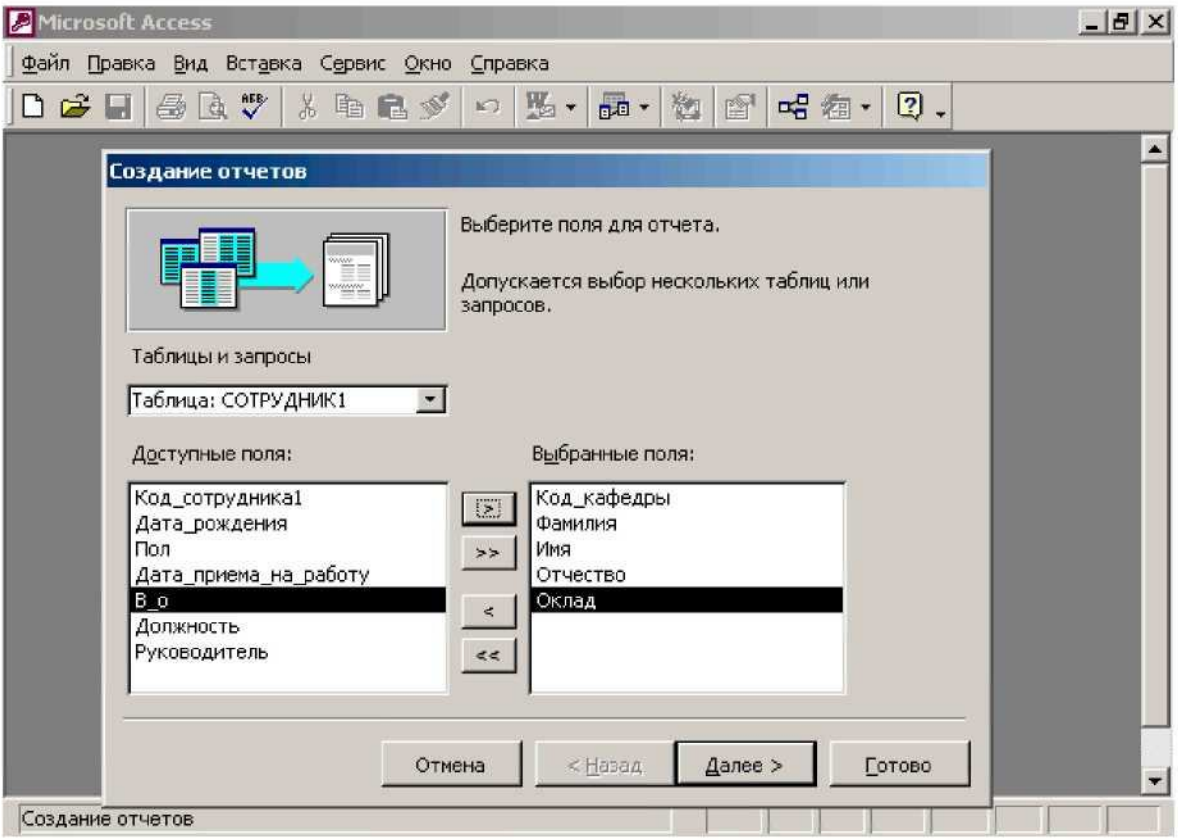


Рис. 2.91. Начальный вид окна «Создание отчетов» при использовании «Мастера отчетов». Выбор полей, включаемых в отчет

Предположим, что мы хотим на основе таблицы «СОТРУДНИК 1» получить ведомость на выдачу зарплаты. Для простоты считаем, что все сотрудники получают фиксированный оклад. Информация сгруппирована по кафедрам. Внутри кафедры записи упорядочены по алфавиту по полям «Фамилия», «Имя», «Отчество».

После определения полей, включаемых в отчет, система прелагает выбрать уровни группировки (рис. 2.92). В нашем примере нас устраивает предложенное поле группировки, и новых уровней группировки вводить не надо. В общем случае, для выделения уровней группировки надо позиционироваться на соответствующее поле, которое будет являться полем, по которому производиться группировка, и нажать на кнопку со стрелкой. Если предполагается несколько уровней группировки, то поля должны выбираться в порядке старшинства группировки.

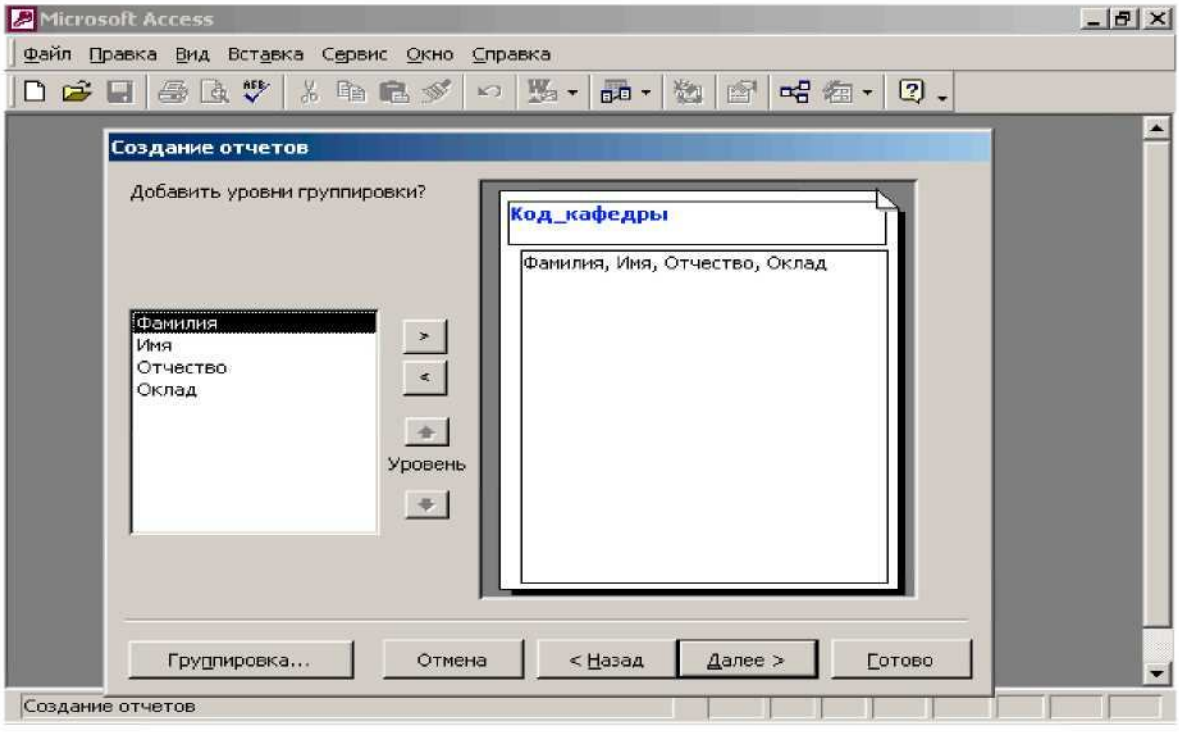


Рис. 2.92. Окно «Создание отчетов». Определение уровней группировки (экран 2)

Следующий экран (рис. 2.93) позволяет задать порядок сортировки и, если необходимо, вычисление итогов (рис. 2.94). Мы выбрали сортировку по трем полям («Фамилия», «Имя», «Отчество») и получение суммарных итогов по полю «Оклад».

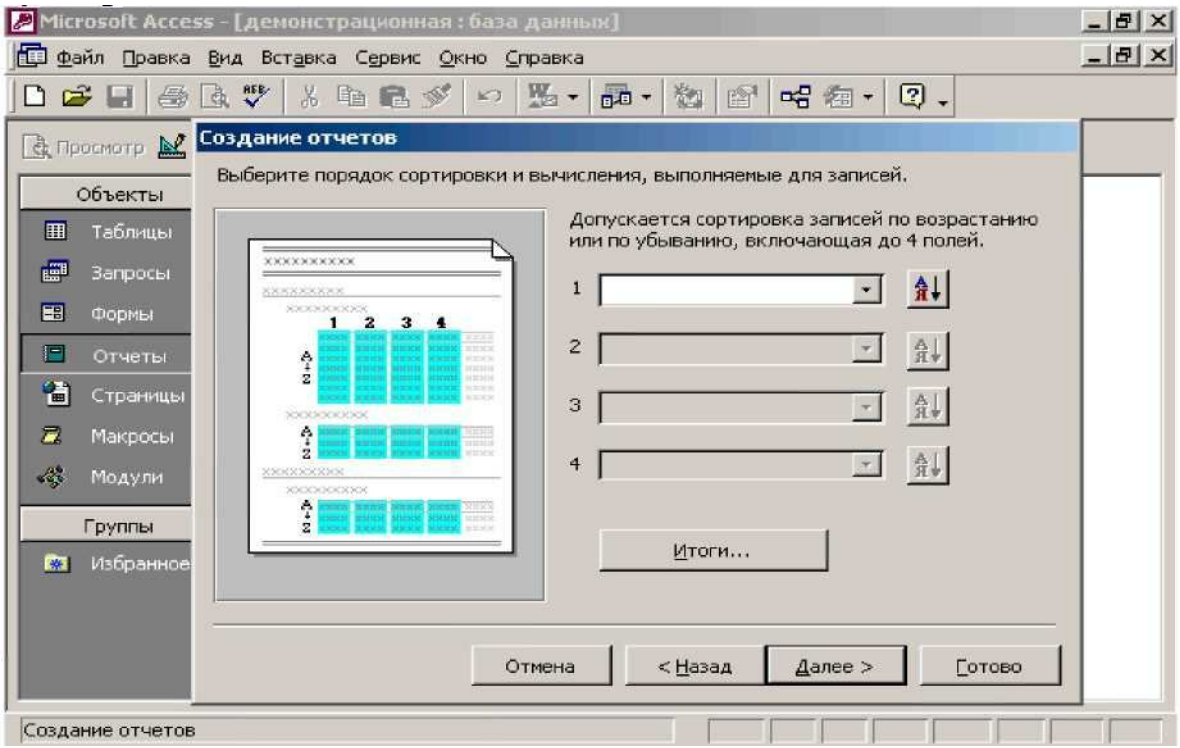


Рис. 2.93. Окно «Создание отчетов». Определение порядка сортировки

Для того, чтобы по каждой кафедре считалась сумма окладов, необходимо нажать кнопку "Итоги" и выбрать соответствующую функцию "Sum".

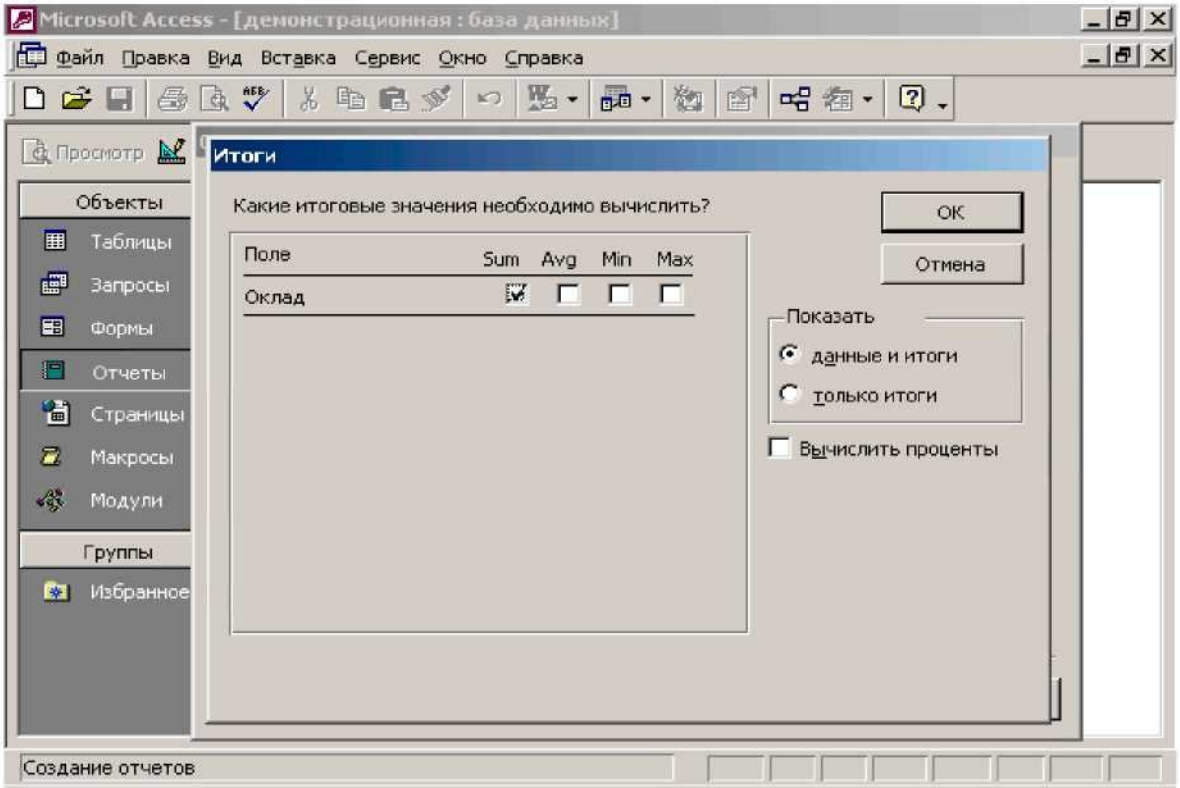


Рис.2.94. Возможность задания итогов

Далее можно выбрать желаемый вид макета отчета и его стиль, за-дать имя отчета (рис.2.95 - 2.97) и нажать на кнопку **"Готово".**

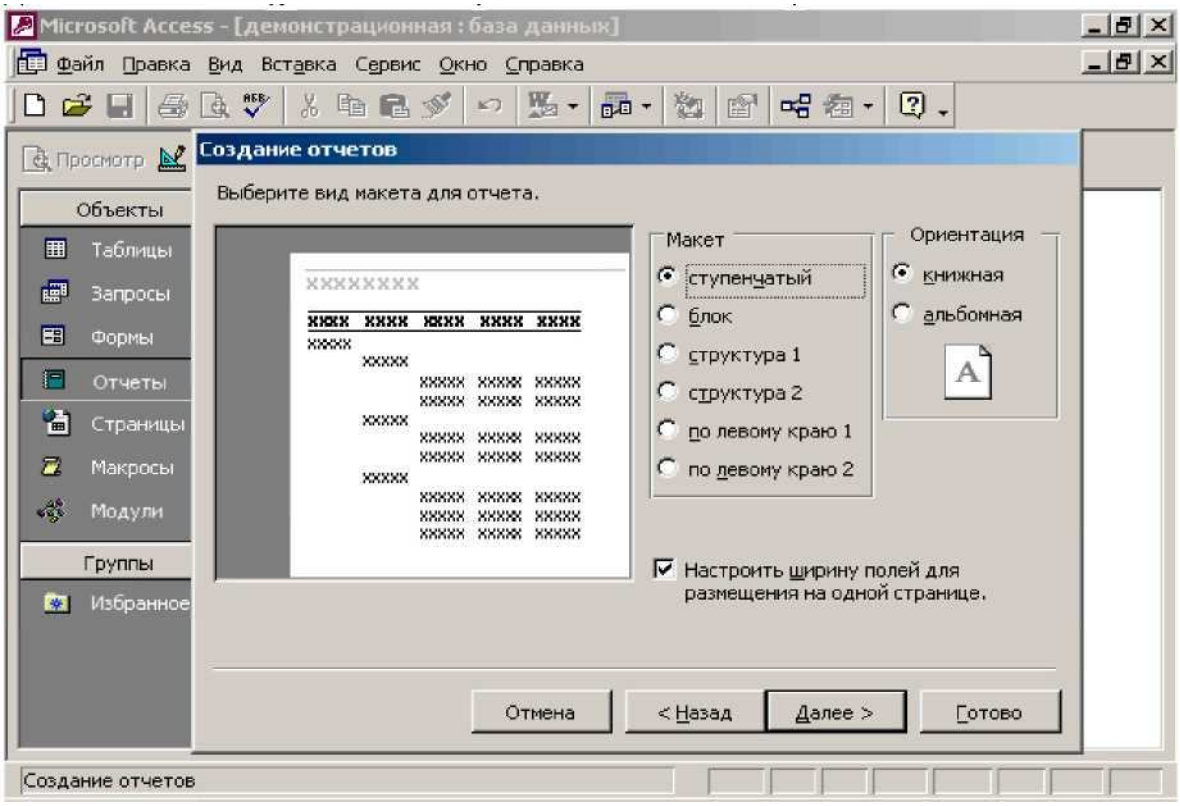


Рис. 2.95. Вид макета отчета

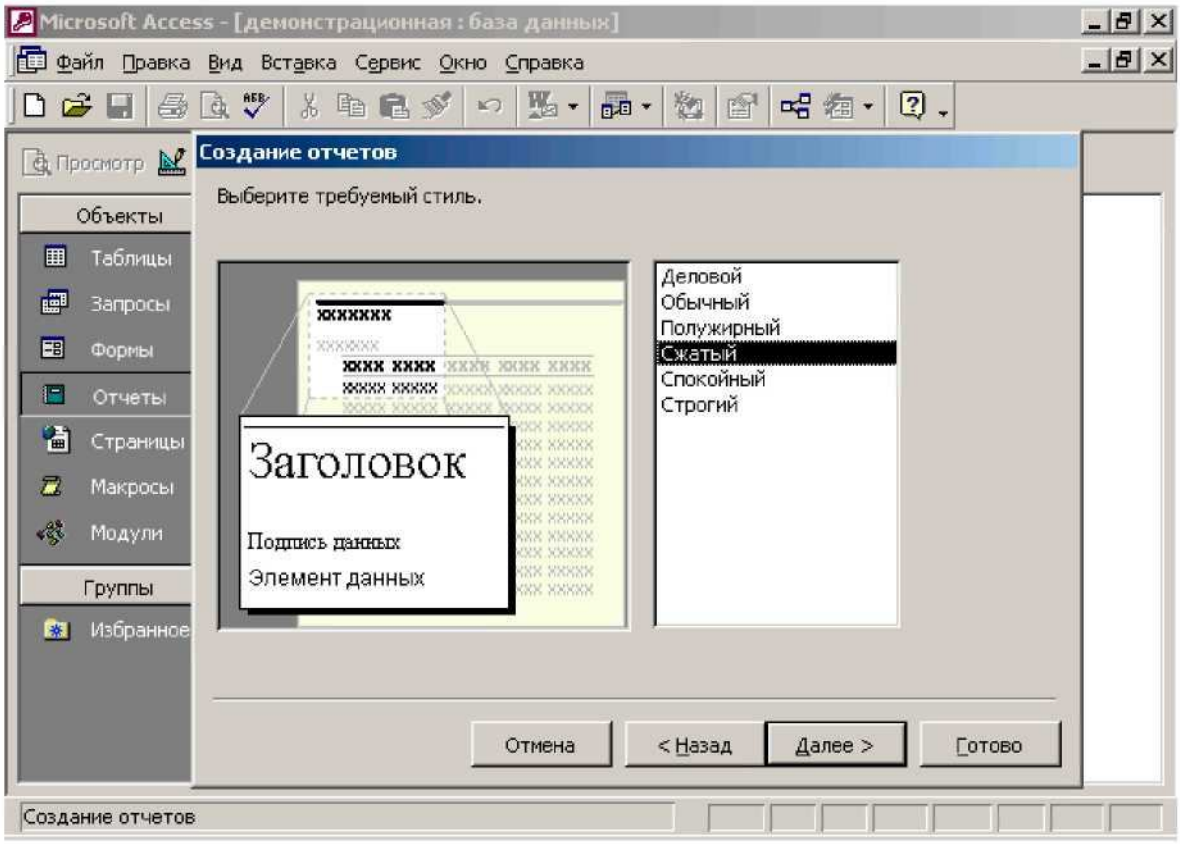


Рис. 2.76. Задание стиля отчета

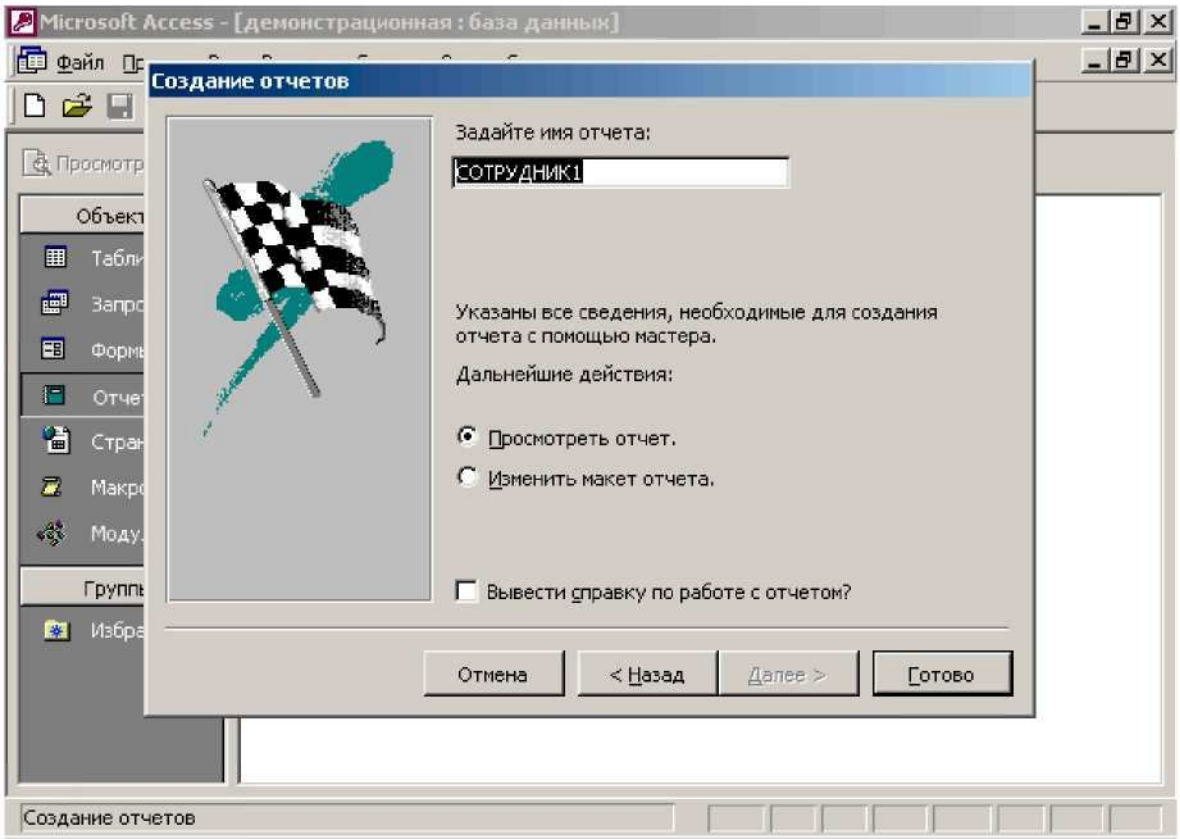


Рис.2.97. Задание имени отчета

В результате выполненных действий будет сформирован отчет. Он будет выведен на экран в режиме предварительного просмотра (рис. 2.98).

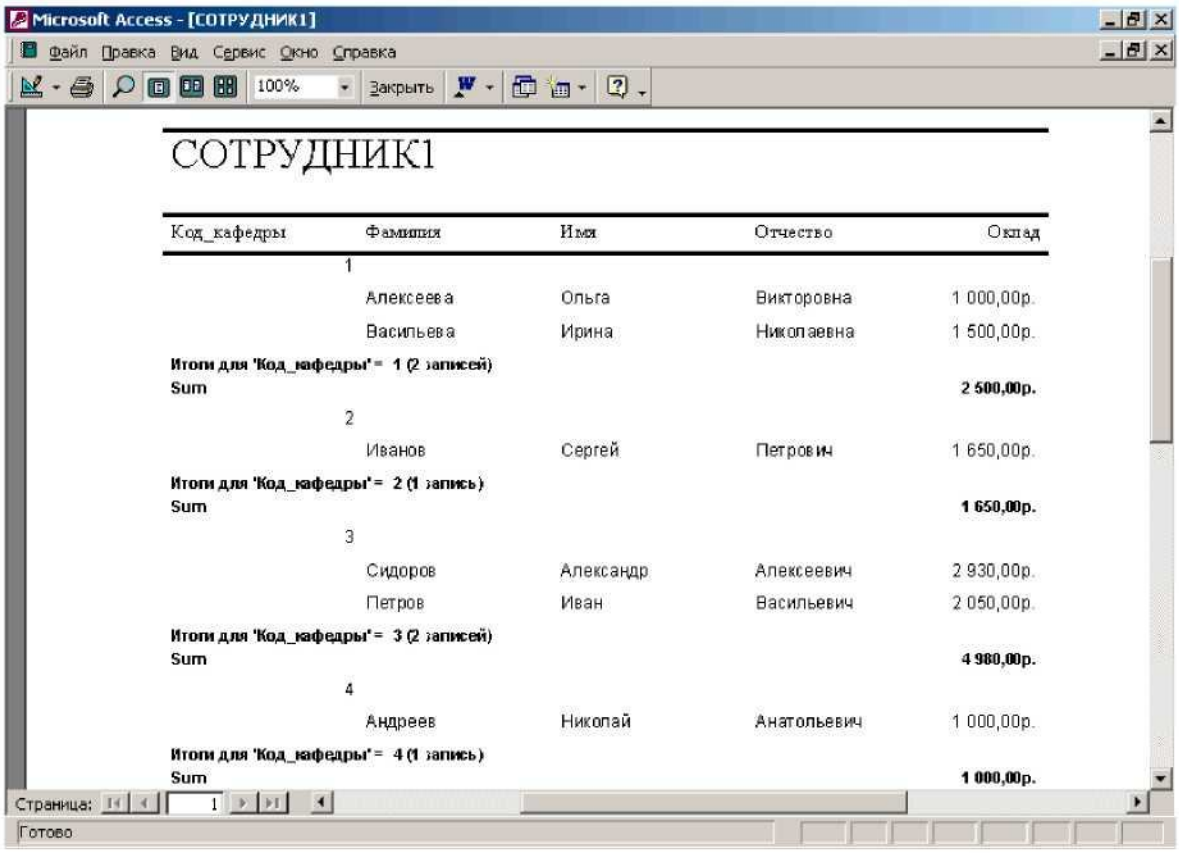


Рис. 2.98. Вид полученного отчета

При создании отчета с использованием «Мастера» на одном из последних шагов система просит задать имя отчета. По умолчанию отчету присваивается имя, совпадающее с именем таблицы, на основе которой формируется отчет (в нашем случае это была таблица «СОТРУДНИК1»). Лучше было бы сразу задать требуемое имя отчета.

Следует обратить внимание, что заданное имя выступает в двух ролях: и имя объекта-отчета, и название документа, выводимое в заголовке отчета. В нашем примере документ будет называться «Ведомость на выплату зарплаты». Такие длинные имена объектам обычно не присваиваются. Если переименовать отчет после его создания, то его заголовок не изменится.

Попробуйте переименовать объект отчет «СОТРУДНИК 1» (задайте ему имя ved_zp). Это можно сделать, например, выделив название отчета в окне базы данных, после чего нажать правую клавишу мыши, в появившемся меню выбрать позицию ***«переименовать»*** и набрать новое имя. Как изменить заголовок отчета, будет рассказано позже.

Кроме того, хотелось бы обратить внимание на то, что в данном разделе мы рассматриваем некоторые возможности генератора отчетов

как инструментального средства и не рассматриваем вопросы проектирования БД. Чтобы примеры были обозримыми реальные ситуации чрезвычайно упрощены. Так, чтобы реально определить зарплату, даже если все сотрудники работают на окладе, надо проверять, полный ли месяц отработал сотрудник (т. е. когда он был принят на работу, не болел ли он или пропускал работу по каким-либо другим причинам, не был ли он в отпуске), не изменялся ли у него в данный период оклад и др. Естественно, что и сами таблицы должны быть спроектированы несколько по иному, чем то, как это сделано в рассматриваемом примере.

Источниками для отчета могут служить несколько таблиц. При определении полей, входящих в отчет, можно последовательно выбирать разные таблицы/запросы и отбирать нужные поля из них (рис. 2.99). Если отчет создается без использования мастера, то при необходимости использовать несколько источников следует сначала создать соответствующий запрос, а потом на его основе формировать отчет.

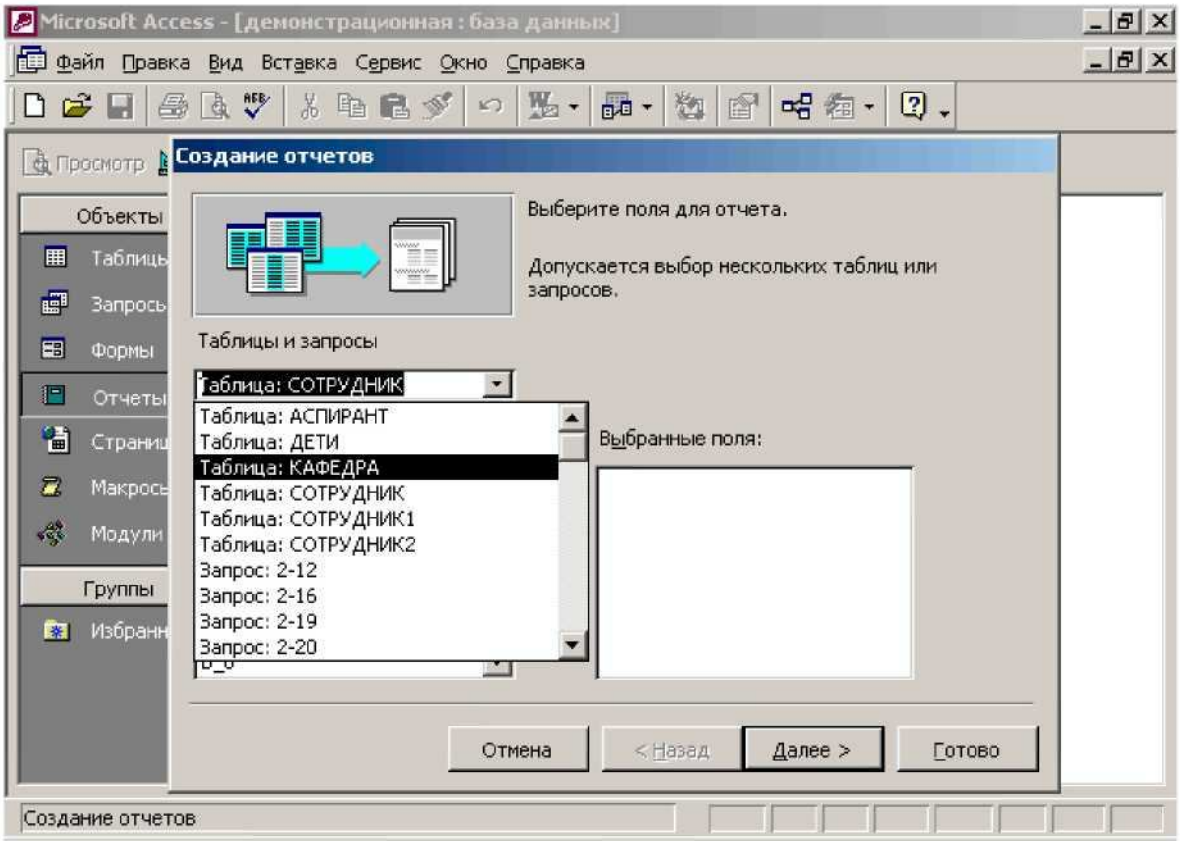


Рис. 2.99. Выбор источников для формирования отчета

Использование кода кафедры в отчете является не очень удобным. Лучше было бы сразу формировать отчет на основе связанных таблиц «КАФЕДРА» и «СОТРУДНИК». Также как и в случае создания экранной формы для связанных таблиц, сначала в качестве источника надо выбрать основную таблицу (в нашем случае это таблица «КАФЕДРА») и поля из нее, а затем - подчиненную («СОТРУДНИК»). Из таблицы

«КАФЕДРА» в отчет следует перенести поле «НАЗВАНИЕ_ КАФЕДРЫ_КРАТКОЕ» или «НАЗВАНИЕ_ КАФЕДРЫ_ПОЛНОЕ»; из таблицы «СОТРУДНИК» - «ФИО» и «ОКЛАД». В этом случае поле группировки система определила бы автоматически.

Если в отчет надо вывести в одной колонке фамилию и инициалы, то следует создать запрос с соответствующим вычисляемым полем (см. гл.2) и использовать этот запрос в качестве источника для формирования отчета.

Создание сложных отчетов

В категорию «Сложных» в Access отнесены: отчеты, включающие в своем составе подчиненные отчеты, перекрестные отчеты, и отчеты, печатающиеся в несколько колонок.

Рассмотрим первую из подкатегорий сложных отчетов. *Подчиненным отчетом* называют отчет, вставленный в другой отчет. При комбинировании отчетов один из отчетов является главным. *Главный отчет* может быть как *присоединенным,* так и *свободным,* т.е. не базирующимся на таблице, запросе или инструкции SQL.

Свободный главный отчет может служить контейнером нескольких не связанных между собой отчетов, которые требуется объединить. Например, вы создаете отчет о продажах за определенный период и хотите включить в него разделы, содержащие данные, сгруппированные по разным признакам: по территориальному, по продуктам, по сотрудникам. Эти разделы будут самостоятельными, несоподчиненными.

Главный отчет связывают с таблицей, запросом или инструкцией SQL в тех случаях, когда в него требуется вставить подчиненные отчеты, в которых выводятся данные, связанные с данными в главном отчете. Например, в главном отчете могут быть выведены записи о каждом сотруднике, а в подчиненном - данные о детях каждого сотрудника. В главный отчет наряду с подчиненными отчетами могут включаться также подчиненные формы, причем число таких подчиненных форм не ограничивается. Более того, главный отчет может содержать подчиненные формы или отчеты двух уровней вложенности. Например, в отчете может содержаться подчиненный отчет, который в свою очередь содержит подчиненную форму или подчиненный отчет.

Подчиненные отчеты могут создаваться либо путем создания подчиненного отчета в существующем отчете, либо путем добавления существующего отчета в другой существующий отчет. В последнем случае добавляемый отчет становится подчиненным.

Если подчиненный отчет должен быть связан с главным отчетом, то перед выполнением следующих действий убедитесь, что правильно установлены связи между соответствующими таблицами.

Создать отчет, включающий подчиненные, можно выполнив следующую последовательность действий:

1. Откройте отчет, который должен быть главным отчетом, в режиме конструктора.
2. Убедитесь, что кнопка **Мастера** на панели элементов нажата.
3. Нажмите кнопку **Подчиненная форма/отчет** на панели элементов.
4. Установите указатель в отчете на том месте, куда требуется поместить подчиненный отчет, и нажмите кнопку мыши. В результате появиться диалоговое окно «Мастер подчиненных отчетов» (рис. 2.100).
5. В появившемся окне выберите таблицу/запрос, являющуюся источником для подчиненного отчета или заранее созданный отчет, который должен быть включен в создаваемый отчет в качестве подчиненного.

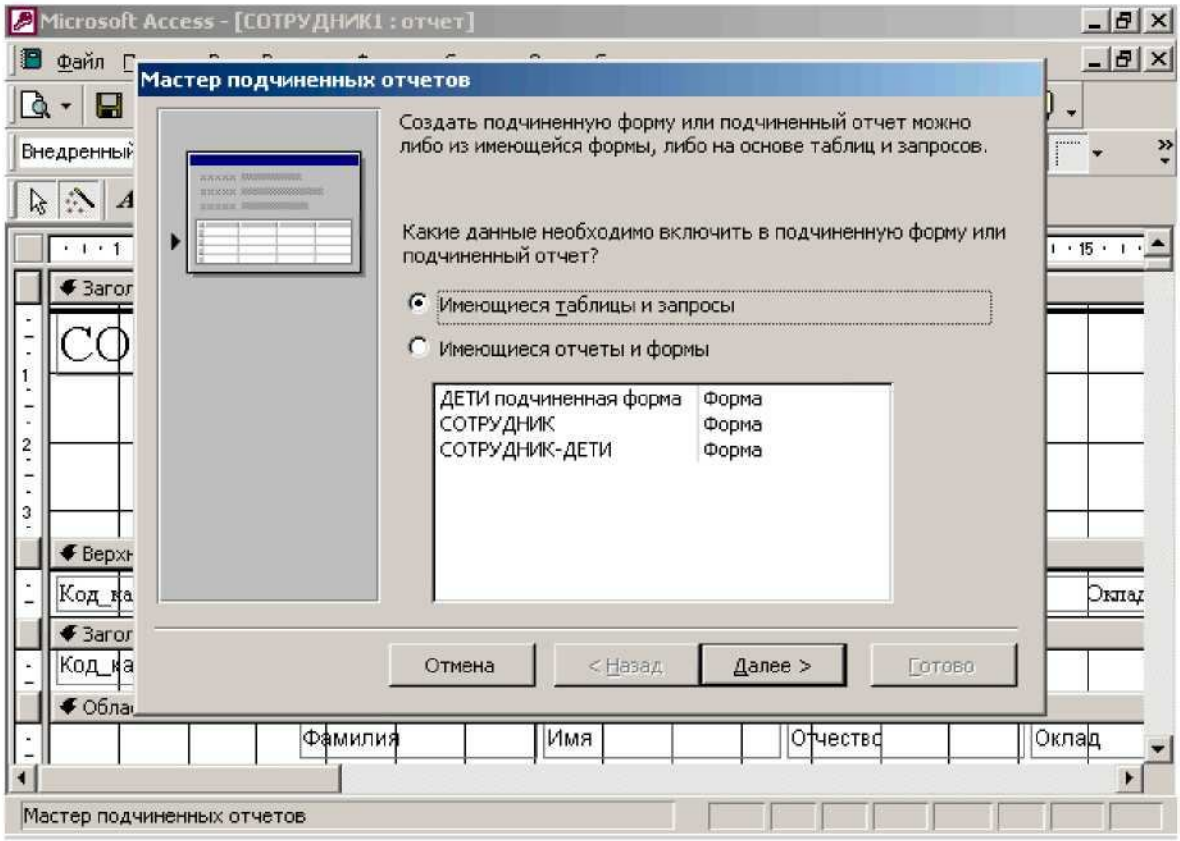


Рис. 2.100. Создание подчиненного отчета

После нажатия кнопки **«Готово»** элемент управления «Подчиненная форма/отчет» будет вставлен в главный отчет. Кроме того, будет создан отдельный отчет, выводящийся как подчиненный отчет.

Создать отчет, включающий подчиненные, можно и другим способом, а именно: после открытия отчета, который должен быть главным отчетом, в режиме конструктора, нажмите клавишу F11 для перехода в окно базы данных и переместите с помощью мыши отчет или таблицу из окна базы данных в тот раздел главного отчета, в который требуется поместить подчиненный отчет.

Сохранение содержания документа

Документ может быть распечатан, либо запомнен в файле. Чтобы запомнить содержимое отчета в текстовом файле, можно использовать позиции меню **«Сервис/Связи с Office/ Публикация в MS Word»,** либо кнопку "Связи с Office", предназначенную для экспорта данных в Word или Excel.

Можно использовать и другой путь для сохранения содержимого отчета, а именно использовать позиции меню: **«Файл/Сохранить как/экспорт» - «во внешнем файле»** и выбрать требуемый тип файла.

Сохранение содержимого отчета в файле следует использовать не только тогда, когда печать отчета и его формирование по каким-то причинам разнесено во времени или «пространстве» (например, отчет должен быть передан в электронном виде пользователю, не имеющего доступа к вашей сети/компьютеру), но и для того, чтобы сохранить отчет в том виде, который соответствовал содержимому базы данных на момент получения данного отчета. Так как при каждом «открытии» отчета его содержимое будет соответствовать состоянию БД на момент формирования отчета.

2.4.2. Корректировка формы отчета

Полученный отчет может в чем-то не соответствовать Вашим потребностям, например, не устраивают стандартные сообщения, выдаваемые по умолчанию, не нужно выводить дату создания отчета, и, наоборот, надо ввести какие-то дополнительные элементы и т. п.

Для того чтобы скорректировать форму отчета, надо перейти в режим конструктора. Для этого можно в пункте меню «Вид» выбрать команду **«Конструктор»** либо выбрать соответствующую кнопку (^ ~) на панели инструментов.

После этого экран будет иметь вид, представленный на рис. 2.101.

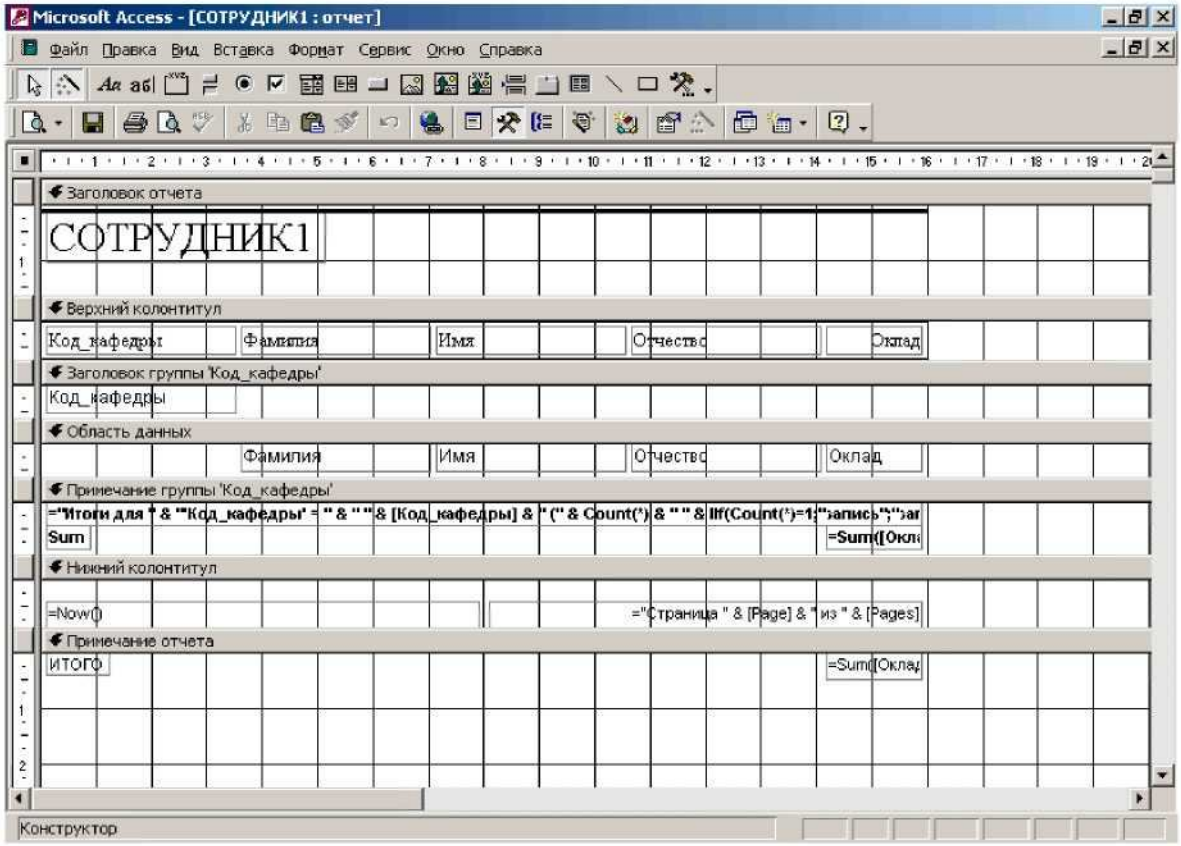


Рис. 2.101. Вид документа в режиме конструктора **Области отчета**

Отчет обычно состоит из нескольких областей. Это области:

- заголовка отчета
- верхнего колонтитула;
- заголовка группы
- области данных
- примечания группы
- нижнего колонтитула
- примечания отчета.

На экране каждая из этих областей ограничивается серой полоской с соответствующим названием.

Если бы мы не использовали возможность задания группировки при формировании отчета, то области заголовка группы и примечания группы отсутствовали бы. Количество областей зависит от выбранного числа уровней группировки.

При использовании автоотчетов все или некоторые из названных областей автоматически включаются в отчет. Если в окне находится меньшее количество областей, чем требуется проектировщику, то в пункте меню "Вид" следует отметить строки, соответствующие нужным

областям отчета. Аналогичным образом ненужные области могут быть удалены из отчета.

Некоторые области могут быть пустыми (не содержать данных).

Данные, находящиеся в области заголовка отчета, выводятся в начале первой страницы отчета либо на отдельной странице. Как правило, в этой области помещают название отчета, логотип фирмы и другие элементы, относящиеся ко всему документу.

Данные, находящиеся в области верхнего колонтитула, выводятся в начале каждой страницы отчета. На первой странице отчета данные из области верхнего колонтитула выводятся вслед за данными из области заголовка отчета. В области верхнего колонтитула размещаются, как правило, названия столбцов, выводимых в отчете табличных данных. Так же в верхнем колонтитуле могут выводиться, например, номера страниц, элементы оформления и другие компоненты, которые пользователь желает размещать в начале каждой страницы документа.

В области данных в самом отчете размещаются значения тех данных, которые необходимо вывести в выходной документ (это могут быть данные из базовой таблицы или запроса или вычисленные в процессе создания отчета показатели). В режиме конструктора в этой области обычно указываются ссылки на поля таблицы (имена полей иногда с указанием полного пути), данные из которых будут выводиться в отчет, формулы для вычисления. В области данных в режиме конструктора можно поместить линии, разграничивающие столбцы/строки отчета, какие-то строковые константы и др. Эти элементы будут в самом отчете повторяться в каждой его строке.

В области нижнего колонтитула размещают элементы, которые хотят выводить внизу каждой страницы отчета (номера страниц могут помещаться как в верхнем, так и в нижнем колонтитуле, некоторые документы требуют наличия заверяющих подписей на каждом листе и т. п.). В рассматриваемом нами примере в конце каждой страницы помещается текущая дата (см. зону «нижний колонтитул» на рис. 2.101), номер страницы и общее число страниц в отчете. На рисунках (2.98, 2.104) эти данные просто не видны, так на них изображена не вся страница, а только ее фрагмент.

В примечании отчета помещают, как правило, итоговые значения по документу в целом (суммы, средние, количества элементов и другие значения, вычисляемые с использованием групповых операций/статистических функций, оформительские реквизиты - подпись, дату формирования документа и т. п.). Данные из этой области выводятся на последней странице отчета.

В режиме конструктора на экране справа и сверху бланка документа можно вывести линейки, помогающие установить размер соответствующего элемента.

Размер любой области документа можно изменять, если подвести указатель мыши к границе области, и, когда он примет вид двусторон-

ней стрелки, при нажатой левой кнопке мыши перетащить границу области в нужном направлении. Следует обратить внимание на то, что размер области данных при использовании табличной формы документа практически означает высоту строки документа.

При работе в режиме конструктора на экране появляются три панели, используемые при создании/корректировке отчетов: «Конструктор отчетов» и «Формат (форма/отчет)» и «Панель инструментов». Каждая из этих моделей может быть отключена. На рис. 2.101 отсутствует панель «Формат». Нижняя панель на этом рисунке является панелью «Конструктора отчетов». Часть из кнопок этого меню является общей для многих офисных Windows-приложений и знакома большинству пользователей. Поэтому рассмотрим далее назначение только тех кнопок, которые являются специфическими, предназначенными именно для построения отчетов:

Ш - кнопка ***"Сортировка и группировка"*** выводит на экран соответствующее окно, в котором указываются поля, используемые для группировки и сортировки данных в отчете (группировка будет описана позднее);

L*L - кнопка ***"Панель элементов"*** включает и выключает режим показа в конструкторе отчетов панели инструментов;

JEJ - кнопка ***"Автоформат"*** выводит на экран диалоговое окно, позволяющее изменить внешний вид всего отчета в целом;

t - кнопка ***"Свойства"*** выводит на экран окно свойств выделенного в данный момент в конструкторе элемента. Каждое поле отчета обладает большим числом свойств. Некоторые из них будут рассмотрены позднее;

^ - кнопка ***"Построить"*** используется для вызова нужного построителя (выражений, макросов или программ);

Элементы отчета

При работе в режиме конструктора появляются панели элементов, отчетов и форматирования. Панель элементов (рис. 2.102) практически не отличается от аналогичной панели, используемой при создании экранных форм. Но, так как отчеты все-таки чаще используются для получения твердых копий документов, такие элементы как поля со списком, списки, переключатели и т. п. элементы используются при создании отчетов редко.

Наиболее часто при создании/корректировке отчетов используются кнопки:

- ***надпись.*** Эти элементы управления попадают в отчет в том виде, в каком они представлены в конструкторе отчетов;

- ***поле.*** В этих элементы управления указываются имена тех полей таблицы или запроса, данные из которых выводятся в отчете;

Любой отчет, также как и форма, включает в свой состав текст, который в неизменном виде выводится в каждом экземпляре документа, поля, которые обеспечивают вывод в документе соответствующих значений из таблиц БД (присоединенные элементы) или вычисленных значений. Кроме того, документы могут включать в себя различные оформительские элементы (линии, прямоугольники, рисунки (например, логотип фирмы) и другую графику). Также существует понятие «свободный» элемент.

Для того чтобы включить в отчет новое поле из БД (т. е. создать присоединенный элемент управления) на панели инструментов следует открыть список полей нажатием одноименной кнопки (). В списке полей выделите одно или несколько полей. Переместите с помощью мыши выбранное поле (или поля) из списка полей в отчет. Поместите верхний левый угол значка в то место, где должен находиться левый верхний угол элемента управления (а не его подписи), и отпустите кнопку мыши.

Кроме того, можно использовать кнопку **«поле»:** нажать ее и перетащить мышью в нужное место отчета. В этом случае создастся свободный элемент, не связанный ни с каким полем таблицы БД. Чтобы «связать» этот элемент, следует выделить его, нажать правую клавишу мыши, в высветившемся меню выбрать позицию **«Свойства»,** а затем на вкладке **«Данные»** нажать кнопку со стрелкой и выбрать поле из списка полей (рис. 2.102).

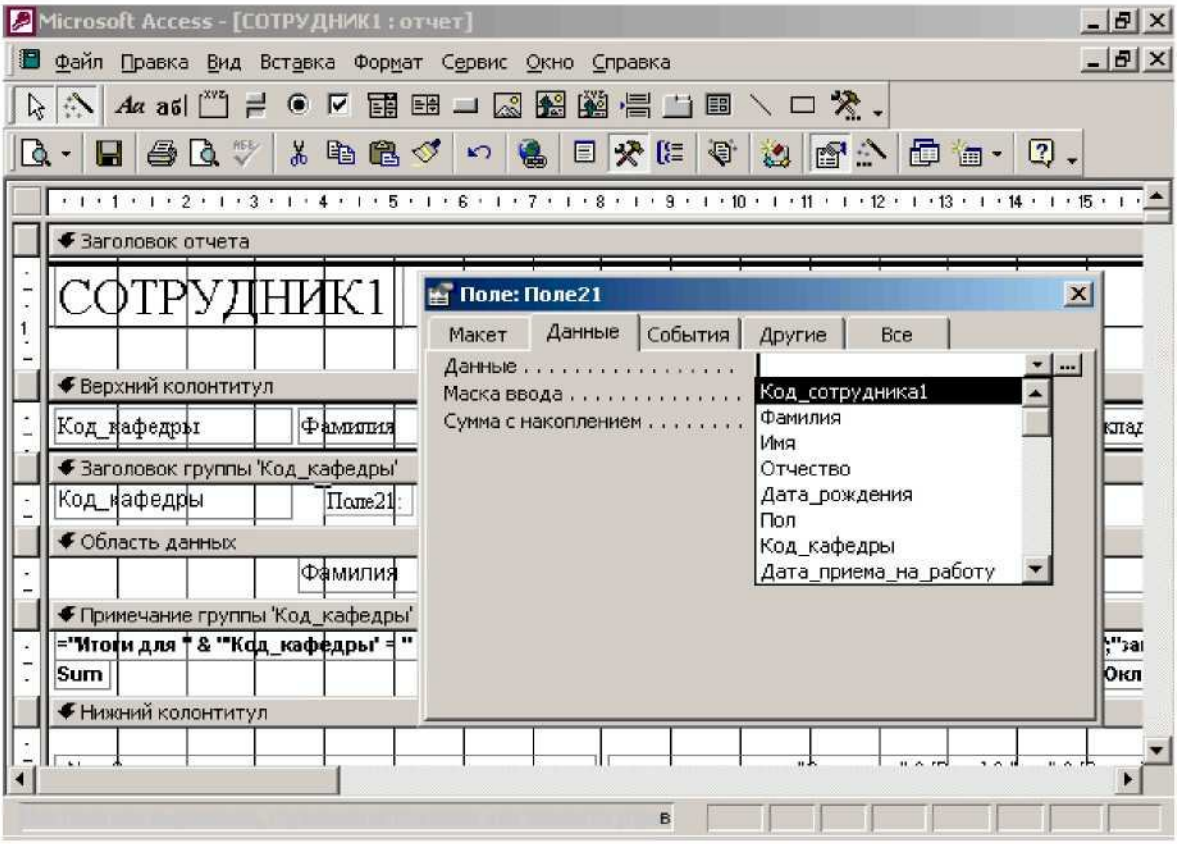


Рис. 2.102. Включение нового поля в отчет. Выбор из списка полей

Чтобы создать вычисляемое поле, в строке **«Данные»** надо записать выражение для его вычисления (лучше воспользовавшись для этих целей построителем, для чего нажать кнопку с многоточием).

Поля выделены в конструкторе отчетов прямоугольниками, показывающими, в каком месте отчета будут выводиться данные, и сколько места отводится для вывода его значения. Местоположение элемента и его размер можно легко менять.

Свойства

Отчет в целом, каждая зона и каждый элемент отчета имеет большой набор свойств. Выйти в окно **«Свойства»** можно позиционировав-шись на соответствующем элементе, щелчком правой кнопки вывести на экран контекстно-зависимое меню и выбрать строку **«Свойства».** При этом на экране появляется соответствующее окно, в котором перечислены все свойства поля. Их можно просматривать по частям, выбирая соответствующие закладки, либо увидеть одновременно, перейдя на закладку **«Все».**

Свойства, собранные на закладке **"Макет",** определяют **как** выводятся данные:

- размещение поля на листе (от левого края, от верхнего края);
- размеры поля (ширина, высота);

- «внешний» вид поля и выводимых в нем данных (тип фона, цвет фона, оформление, тип границы, цвет границы, ширина грани цы, цвет текста, шрифт, размер шрифта, насыщенность, курсив, под черкнутый, выравнивание текста);

- способ представления данных в поле (формат поля, число десятичных знаков, вывод на экран, расширение, сжатие).

Изменение свойств производится путем выбора из раскрывающегося списка нужного значения. Большинство свойств очевидно и не требует каких-либо пояснений.

Свойства на закладке **«Данные»** определяют, что выводится в поле. Использование этих свойств («Данные» и «Сумма с накоплением») мы уже демонстрировали выше.

Свойства, собранные на закладке **"Другие"** (имя и дополнительные сведения) используются соответственно для задания имени поля и задания примечаний, относящихся к нему.

Если требуется изменить свойства нескольких полей, то их выделяют при нажатой левой кнопке мыши, а затем указывают необходимые свойства. Они распространяются на все выделенные поля.

Откорректируем отчет, изображенный на рис. 2.98. Прежде всего, изменим заголовок отчета. Для этого нажмем кнопку **«Надпись»,** и вместо названия *«Сотрудник»* напишем *«Ведомость на выдачу зарплаты»*.

Далее изменим выражение:

="Итоги для " & "'Код_кафедры" = "&""& [Код_кафедры] & " (" & Count(*) & " " & IIf(Count(*)=1;"запись";"записей") & ")",

Sum =8шп([Оклад])

записанное в зоне ***«Примечание группы Код_кафедры»,*** на

="Итоги для кафедры " & [Код_кафедры]

=Sum([Оклад])

Тем самым мы откорректировали подписи и отказались от подсчета числа записей в группе. Отчет в режиме конструктора теперь имеет вид, представленный на рис. 2.103.

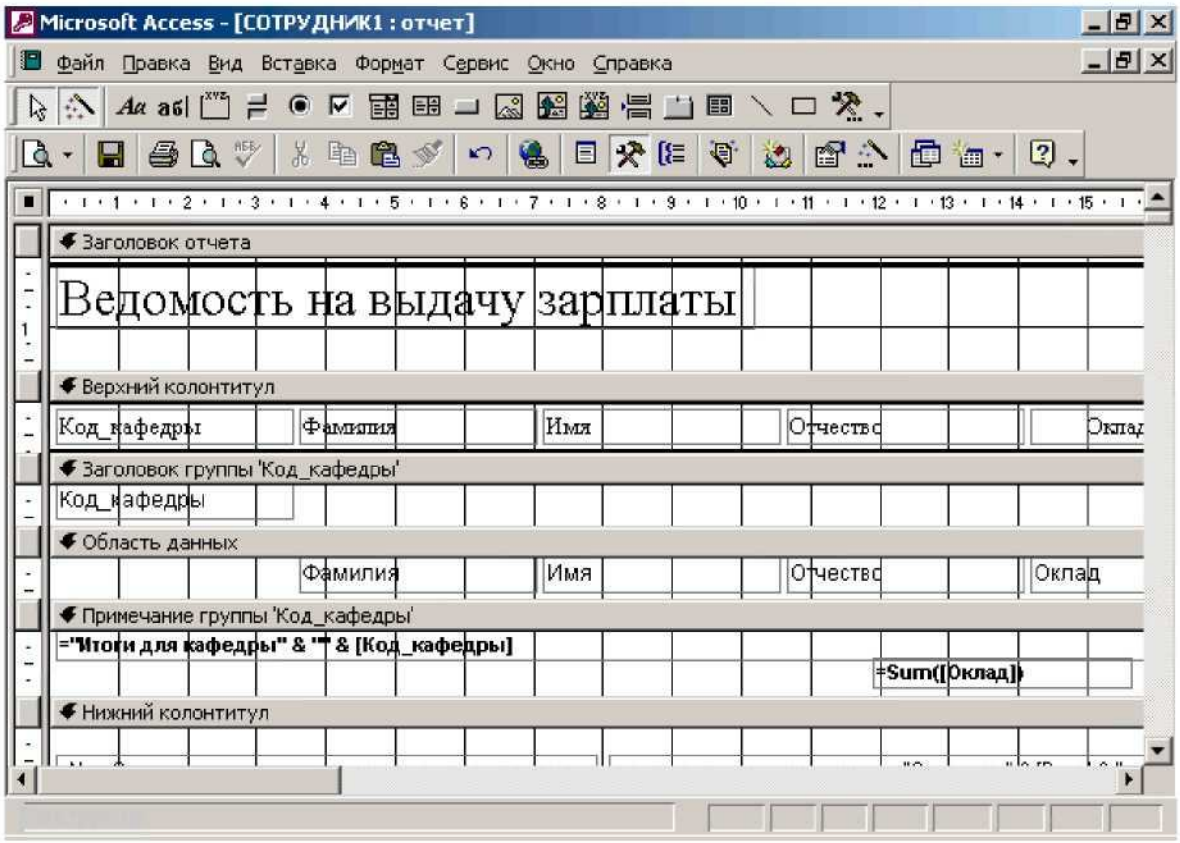


Рис. 2.103. Вид скорректированного отчета в режиме конструктора

Отчет в режиме просмотра имеет вид, представленный на рис. 2.104.

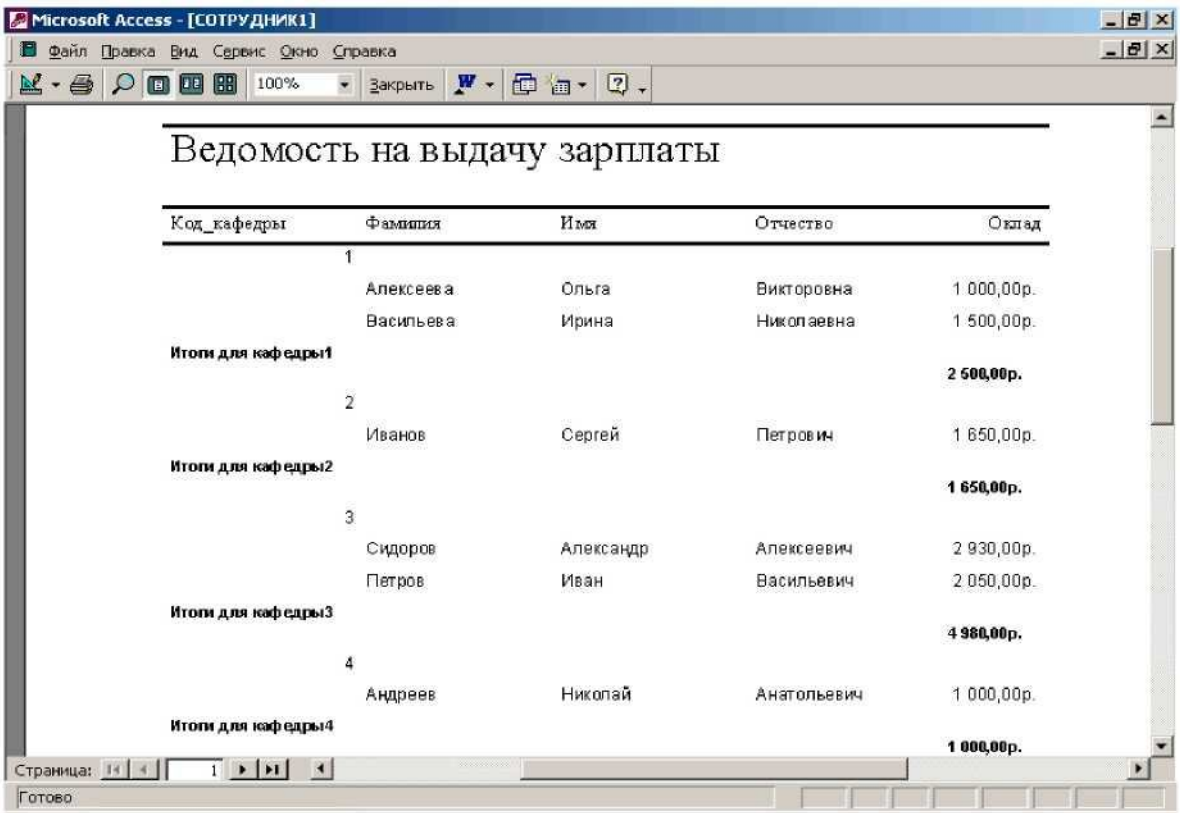


Рис. 2.104. Вид скорректированного отчета в режиме предварительного просмотра

Продолжим корректировку полученного нами отчета, демонстрируя тем самым другие возможности генератора отчетов. Прежде всего, мы хотим перенести название поля «Код кафедры» в зону *Заголовок группы «Код_кафедры».* Для этого надо активизировать данный элемент отчета, подведя указатель мыши к любому месту выбранного элемента и один раз щелкнуть левой кнопкой мыши (вокруг активных элементов появляются маркеры - маленькие черные квадратики по углам и в центре каждой из сторон)., после чего добиться, чтобы указатель мыши принял форму «ладошки», нажать правую клавишу мыши, не отпуская ее, перенести элемент на нужное место (предварительно передвинув вправо элемент-поле «Код_кафедры» в этой зоне чуть правее, чтобы освободить место для расположения надписи).

После этого изменим размер обоих элементов в зоне заголовка группы, чтобы они отображались более компактно. Изменить размер элемента можно, выделив его. После чего следует позиционировать указатель мыши так, чтобы он принял форму двунаправленной стрелки, нажать левую клавишу мыши и перетащить стрелку в нужном направлении до достижения требуемого размера элемента.

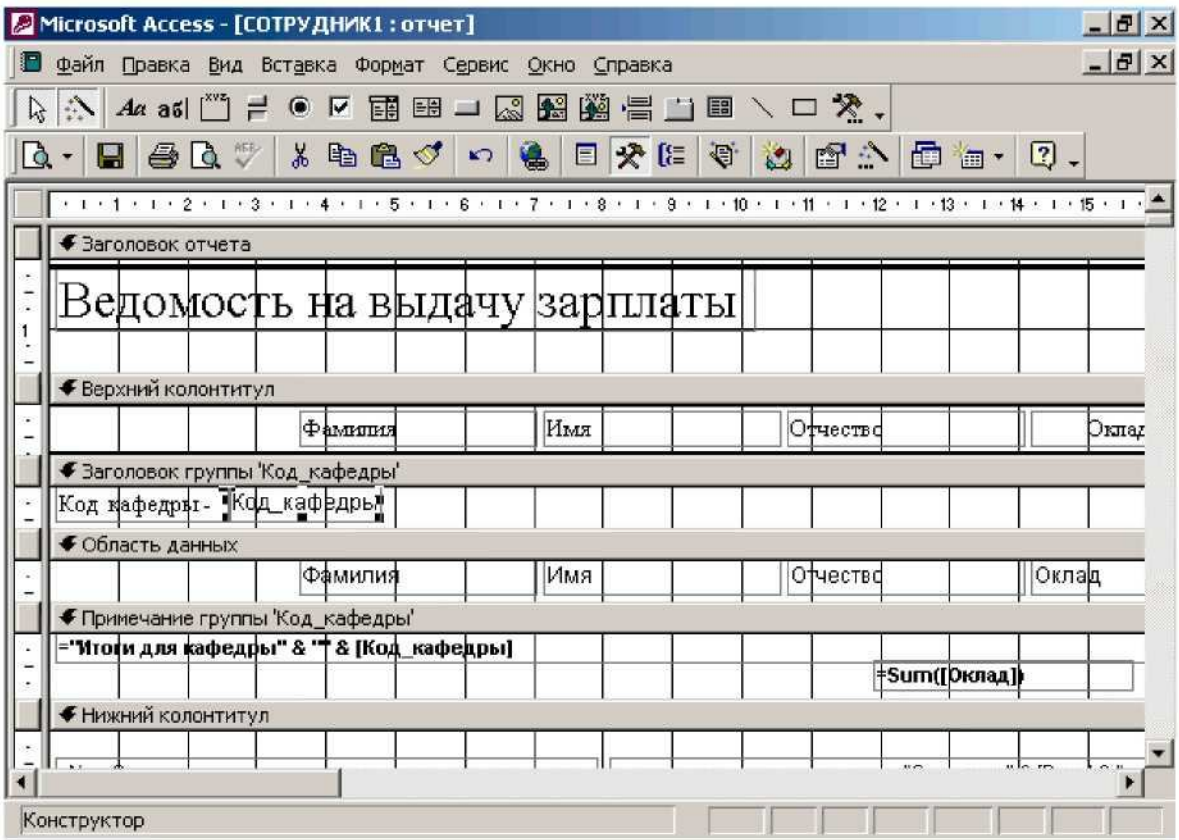


Рис. 2.105. Вид скорректированного отчета в режиме конструктора

Наша форма в режиме конструктора теперь имеет вид, представленный на (рис. 2.105), а в режиме предварительного просмотра - на рис. 2.106.

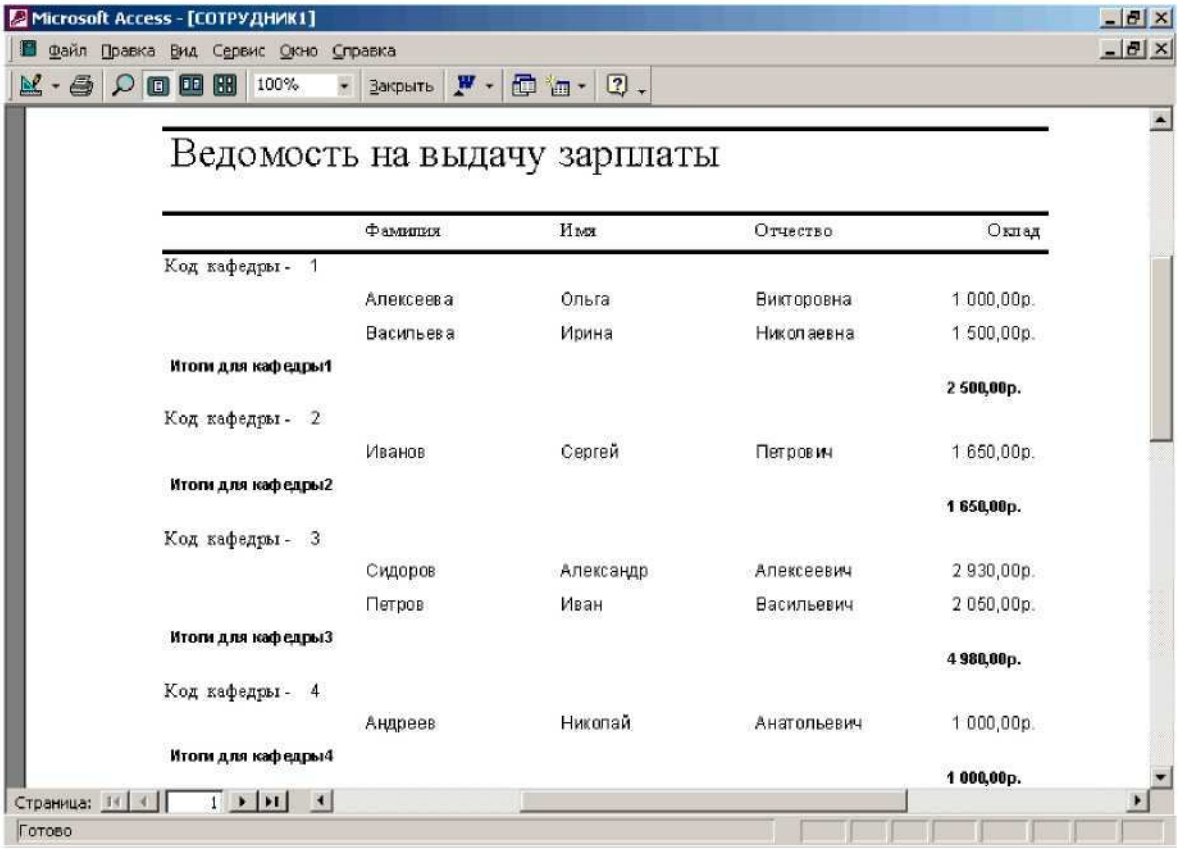


Рис.2.106. Вид скорректированного отчета в режиме предварительного просмотра

Вычисления в отчете

В отчетах (также как и в запросах, формах) можно использовать вычисляемые поля. Мы уже при рассмотрении отчета, полученного с использованием мастера, встречались с ними и даже корректировали их (речь идет об итоговых показателях, номерах страниц, дате). Теперь рассмотрим, как можно создавать вычисляемые поля.

Так как вычисляемое поле, как следует из его названия, является полем, то для его создания в отчет следует включить элемент-поле (при этом в *"области данных"* создается поле, внутри которого вместо имени поля таблицы указано слово *"Свободный")*. Чтобы ввести выражение для вычисления значения поля следует выделить это поле, щелчком правой кнопки мыши вывести на экран контекстно-зависимое меню, выбрать в нем строку **«Свойства»,** в появившемся окне свойств поля перейти на позицию **«Данные»** и в данной строке ввести требуемое выражение. Оно может вводиться «вручную» либо строиться с использованием по-строителя выражений.

Выражение, вводимое в поле, должно начинаться со знака равенства. Для того чтобы воспользоваться построителем, следует нажать на кнопку с многоточием. Построение выражение выполняется как обычно.

Предположим, что мы хотим в нашу ведомость ввести графу «Подоходный налог» (для простоты будем считать, что все сотрудники платят налог в размере 12%). Выражение, записанное в строку «Данные» окна «Свойства полей», будет иметь следующий вид:

= [Оклад] * 0,12.

Чтобы закончить оформление вводимой графы, в *"зону верхнего колонтитула"* введем (воспользовавшись кнопкой **«Надпись»)** название этой графы «Подоходный налог»). Ведомость в режиме конструктора теперь имеет вид, представленный на рис. 2.106.

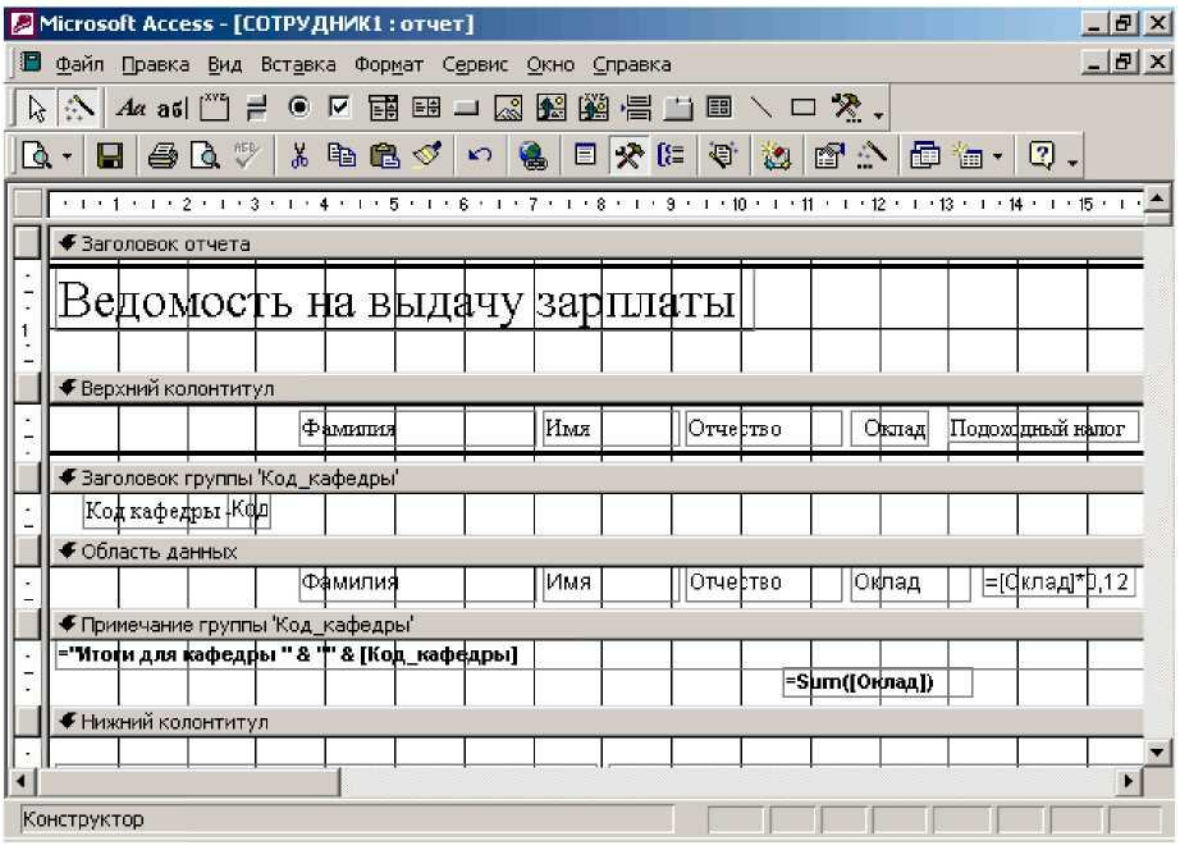


Рис. 2.106. Вид скорректированного отчета в режиме конструктора

Ведомость в режиме просмотра имеет вид, представленный на рис.2.107

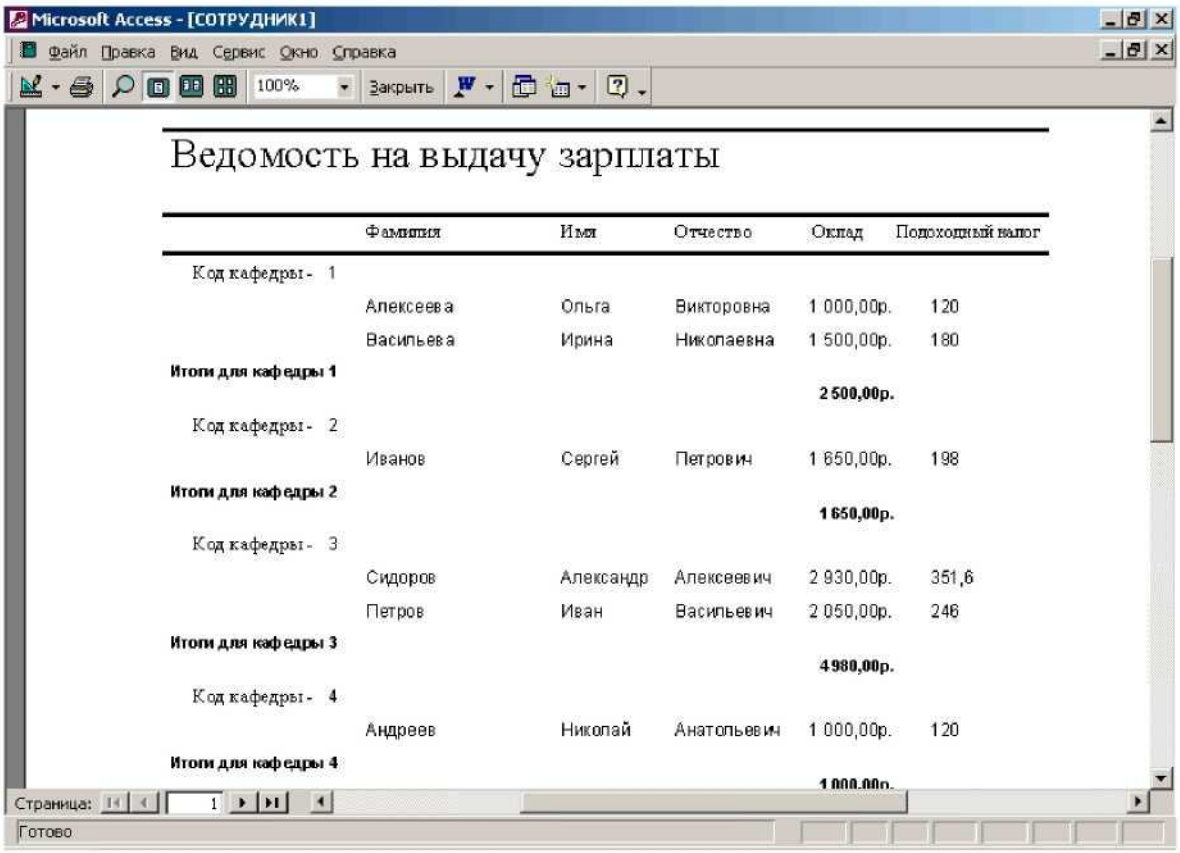


Рис. 2.107. Вид скорректированного отчета в режиме предварительного просмотра

Часто в отчете требуется нумеровать записи. Для этого в режиме конструктора отчета надо добавить в область данных новое поле. Затем установить на него указатель и открыть окно его свойств двойным нажатием кнопки мыши. В ячейку свойства **«Данные»** ввести выражение =1 (рис. 2.108). В ячейке свойства **«Сумма с накоплением»** выберите значение **«Для всего»,** если вы хотите, чтобы нумерация была сплошной, или **«Для группы»,** если хотите, чтобы нумерация была в пределах группы. Мы выбрали для нашего примера последнюю из перечисленных возможностей. Далее в зоне верхнего колонтитула введем имя колонки «№ п/п».

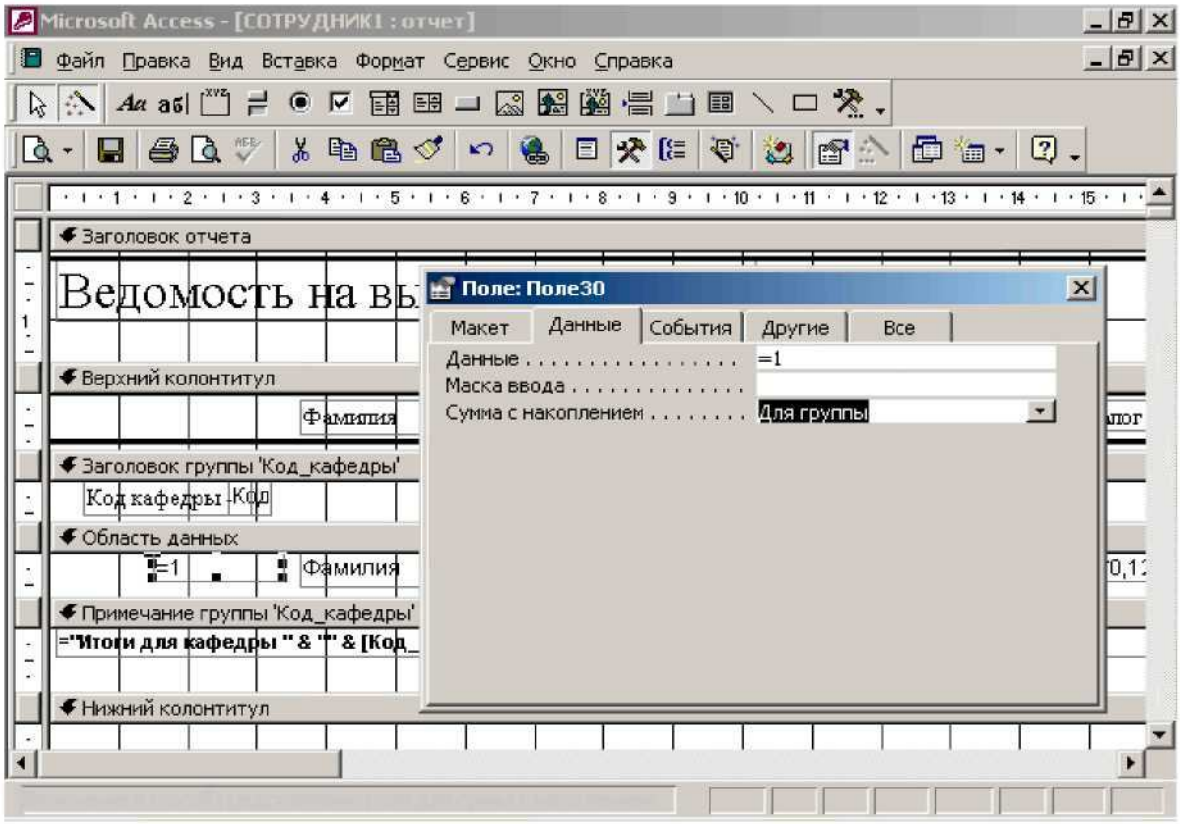


Рис. 2.108. Экран свойств (задание номера по порядку)

В результате проделанных операций отчет будет иметь следую-щий вид (рис. 2.109).

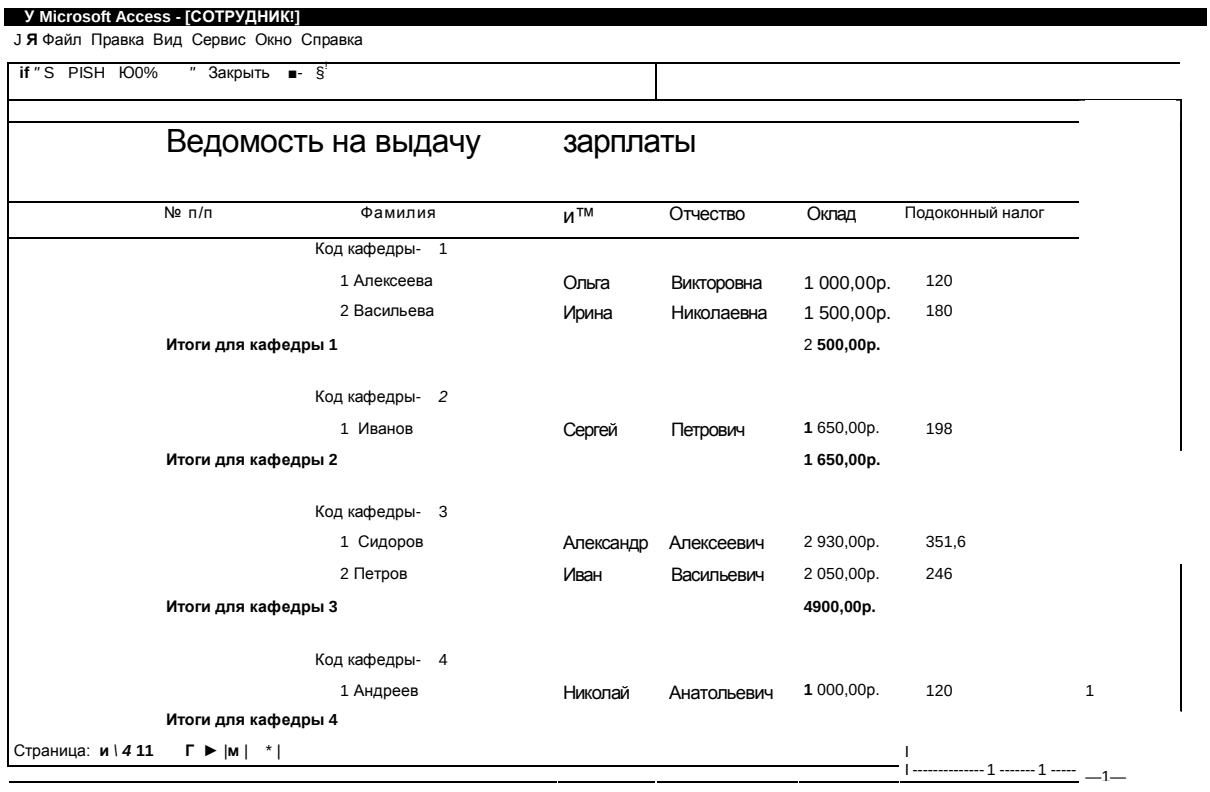


Рис. 2.109. Вид скорректированного отчета в режиме предварительного просмотра

Группировка

При создании отчета с помощью мастера был задан один уровень группировки - по коду кафедры. Предположим, что мы хотим создать еще один уровень группировки - по полу, и сосчитать среднюю зарплату по этой группе. Для задания сортировки и группировки следует воспользоваться соответствующей кнопкой (). Появится окно «Сортировка и группировка» (рис. 2.110). В графе «Поле/выражение» надо из ниспадающего списка выбрать поле, по которому производится упорядочение или группировка. Чтобы по полю производилась группировка, надо задать значение «да» для свойств «Заголовок группы» и/или «Примечание группы». Слева от поля группировки появляется соответствующий знак.

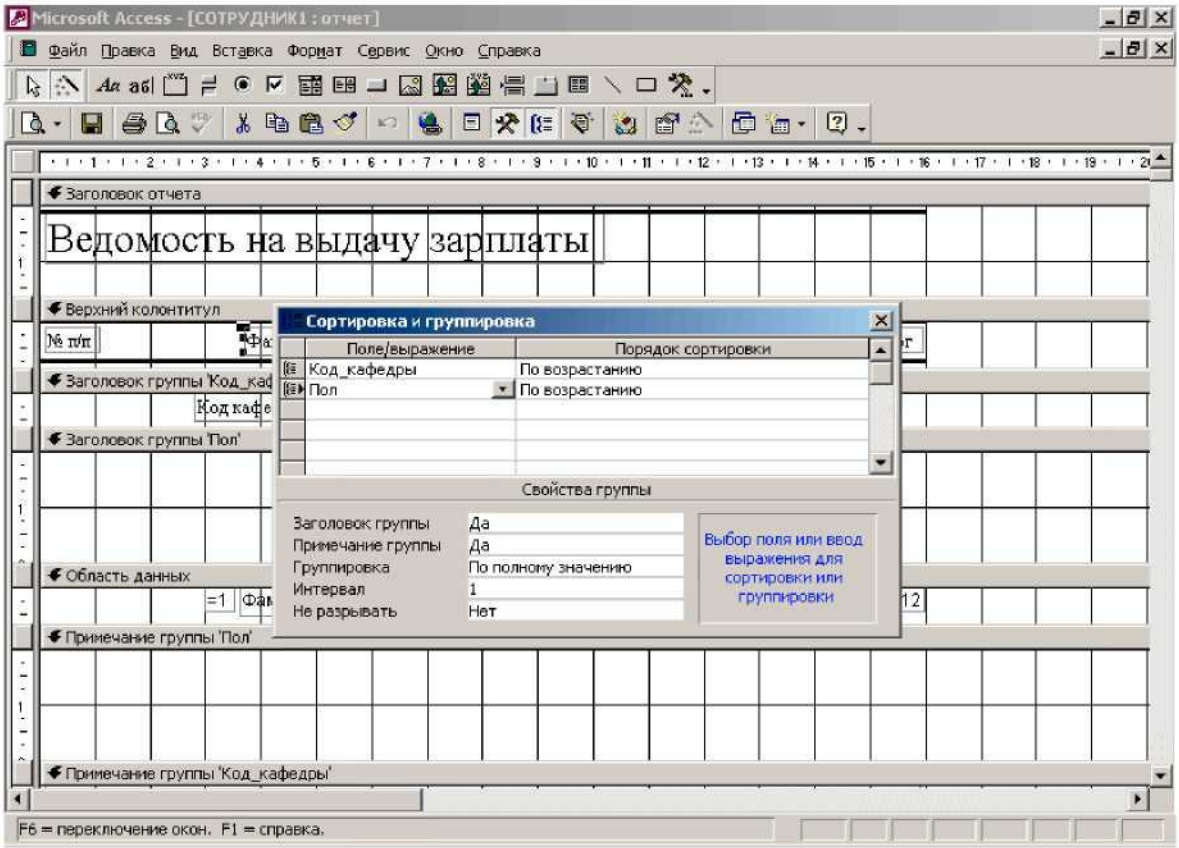


Рис. 2.110. Введение уровней группировки и сортировки

В примечание группы «Пол» включаем новое поле. В окне свойств этого элемента зададим подпись «Средняя зарплата», и, воспользовавшись построителем, в строку данные введем формулу для вычисления средней зарплаты. Вид скорректированного отчета в режиме конструктора представлен на рис. 2.111.

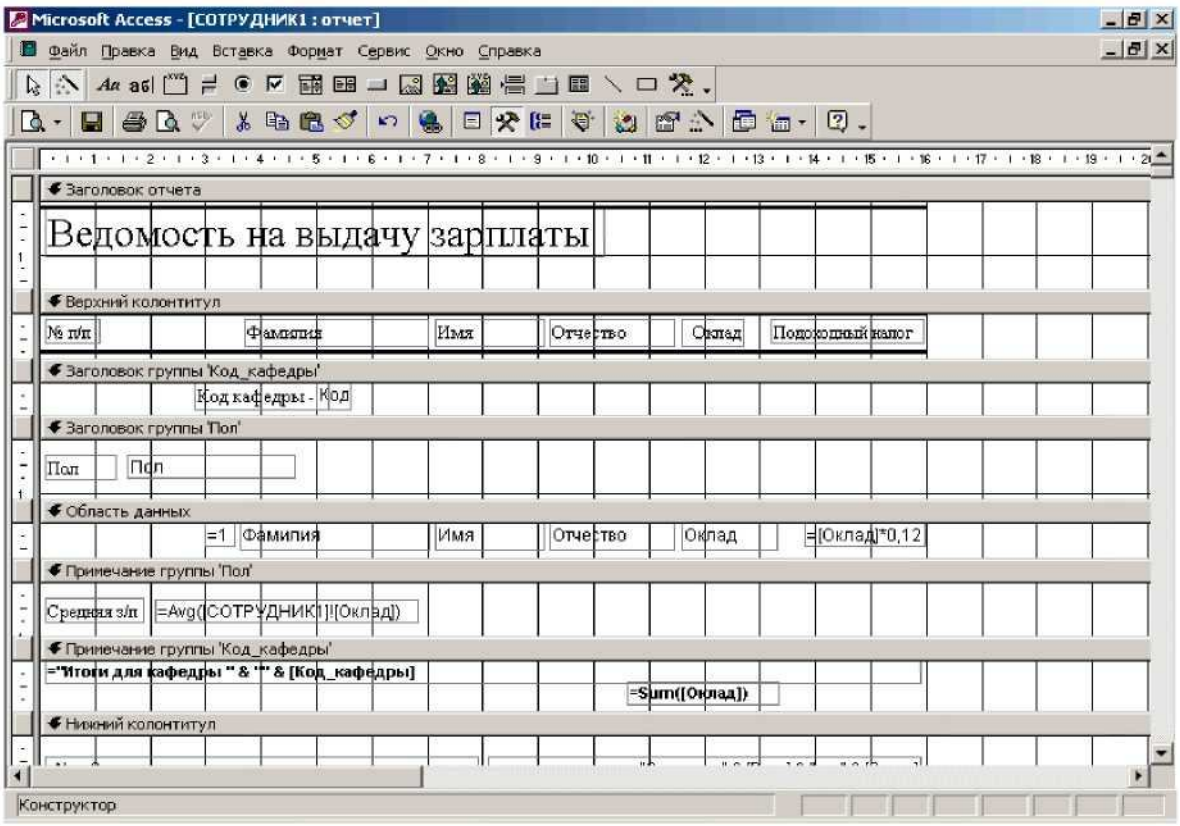


Рис. 2.111. Вид скорректированного отчета в режиме конструктора

В результате документ будет иметь вид, представленный на рис. 2.112.

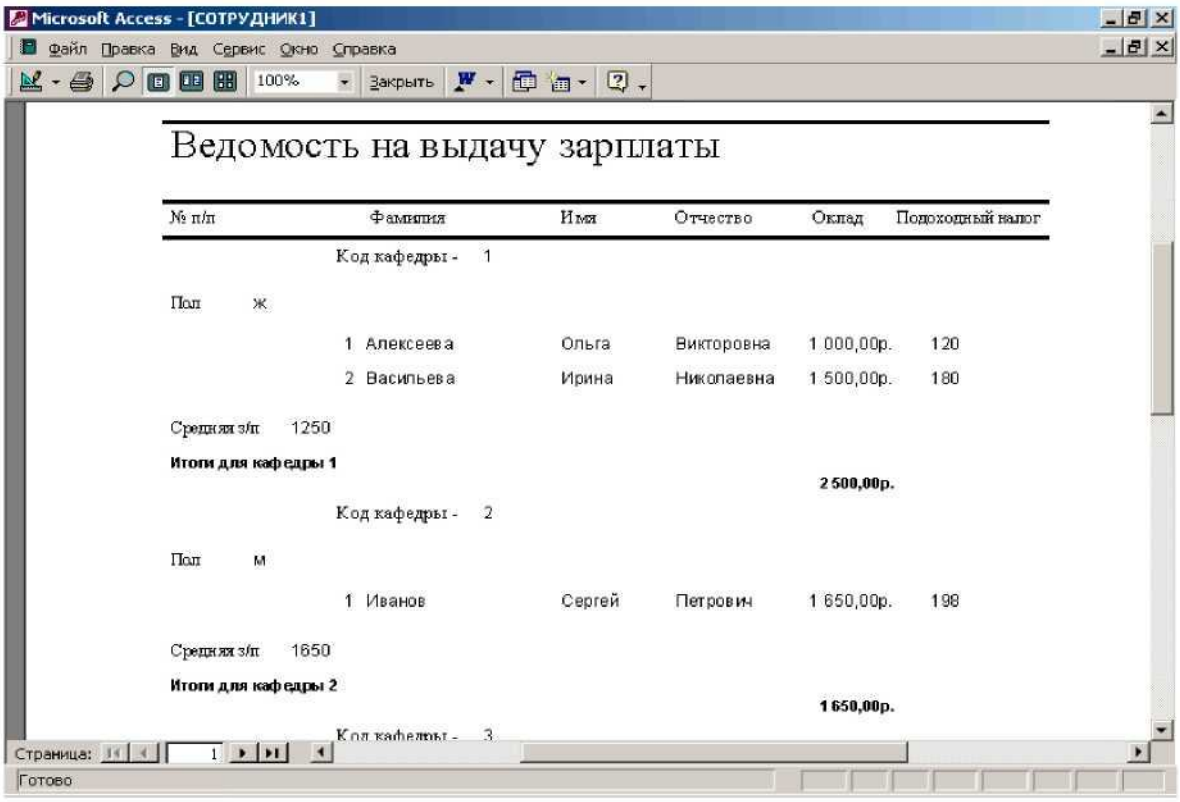


Рис. 2.112. Фрагмент отчета в режиме предварительного просмотра после введения дополнительного уровня группировки

Всего в отчете может быть задано до 10 уровней группировки.

Задание номеров страниц

В многостраничных документах возникает необходимость нумеровать страницы. Это можно сделать, выбрав позиции меню **«Вставка/Номер страницы»** и в появившемся окне (рис. 2.113) выбрать формат и размещение номера страницы.

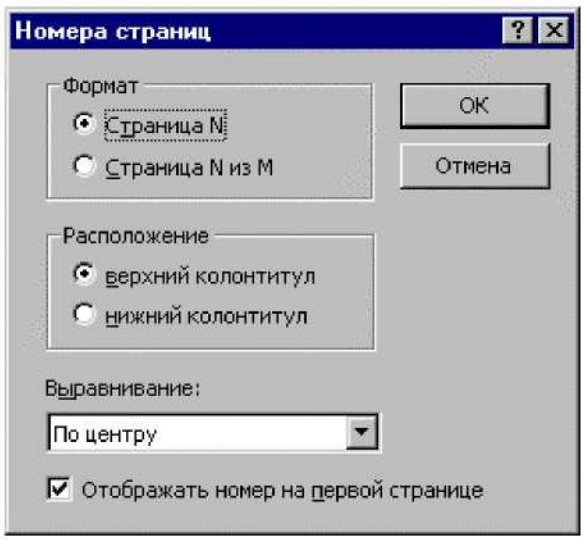


Рис. 2.113. Окно диалога «Номера страниц»

Использование графических элементов

Для улучшения внешнего вида в отчете можно использовать графику: линии, прямоугольники, графические изображения. Чтобы использовать в отчете соответствующие элементы, на панели инструментов следует щелчком мыши выделить соответствующую кнопку, а затем, при нажатой левой кнопке мыши, указать, где будут располагаться левый верхний и правый нижний угол создаваемого изображения. Отпустив кнопку, вы увидите на экране нужный элемент. Попробуйте, используя эти элементы, оформить свой документ.

Кнопки, предназначенные для вставки в отчет графических изображений, используются аналогично кнопкам **«Линия»** или **«Прямоугольник».** Отличие состоит в том, что после определения прямоугольной области, определяющей местоположение рисунка, запускается диалог, в процессе которого выбирается файл, из которого будет производиться вставка рисунка.

Параметрические отчеты

Иногда в документы необходимо вставлять данные, которые отсутствуют в таблицах БД и не могут быть вычислены. Одним из спосо-

бов осуществления этого является использование «параметрического» отчета. Примером такой ситуации является необходимость печатать в заголовке Ведомости на выдачу зарплаты название месяца. Это можно сделать следующим способом: вставить в соответствующее место отчета новое поле, после этого войти в свойства этого элемента и в строке данные ввести слово «месяц» (рис. 2.114), а в строке подпись - «за» (рис. 2.115).

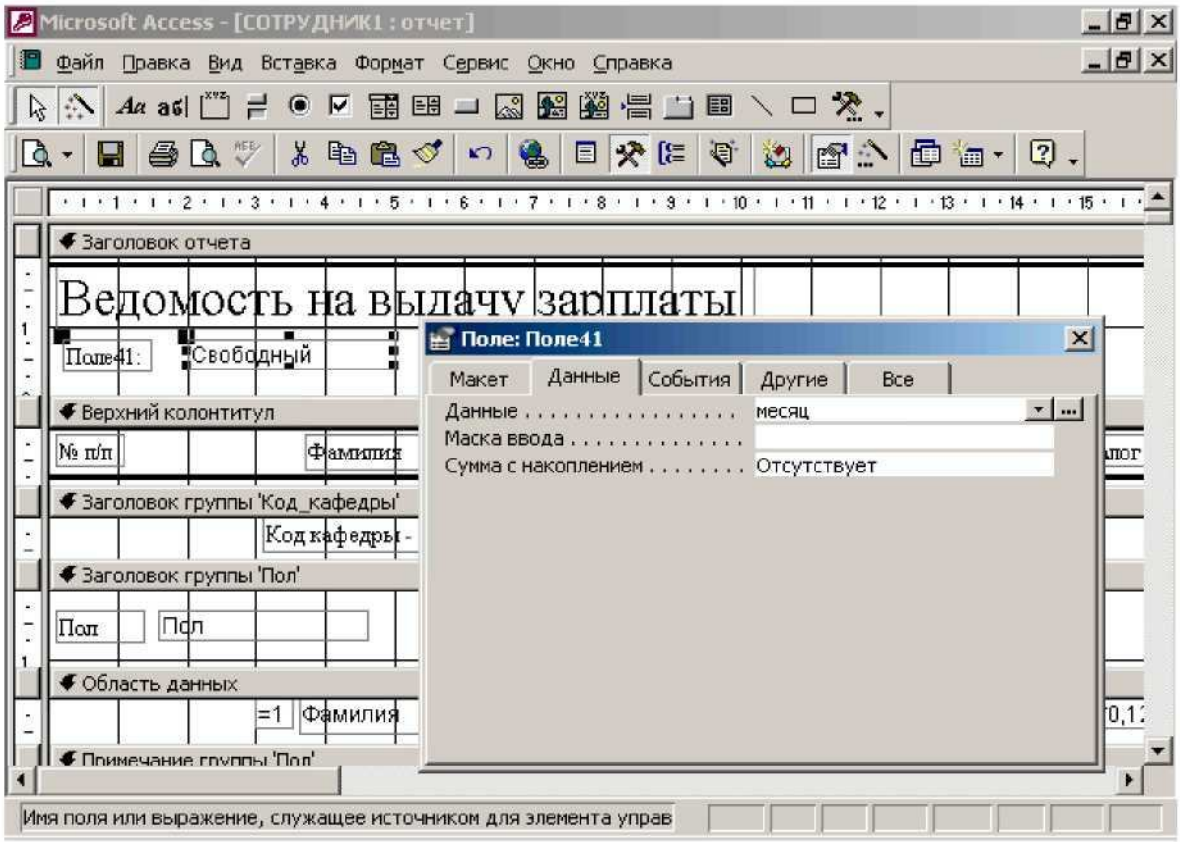


Рис. 2.114. Создание отчета с параметрами

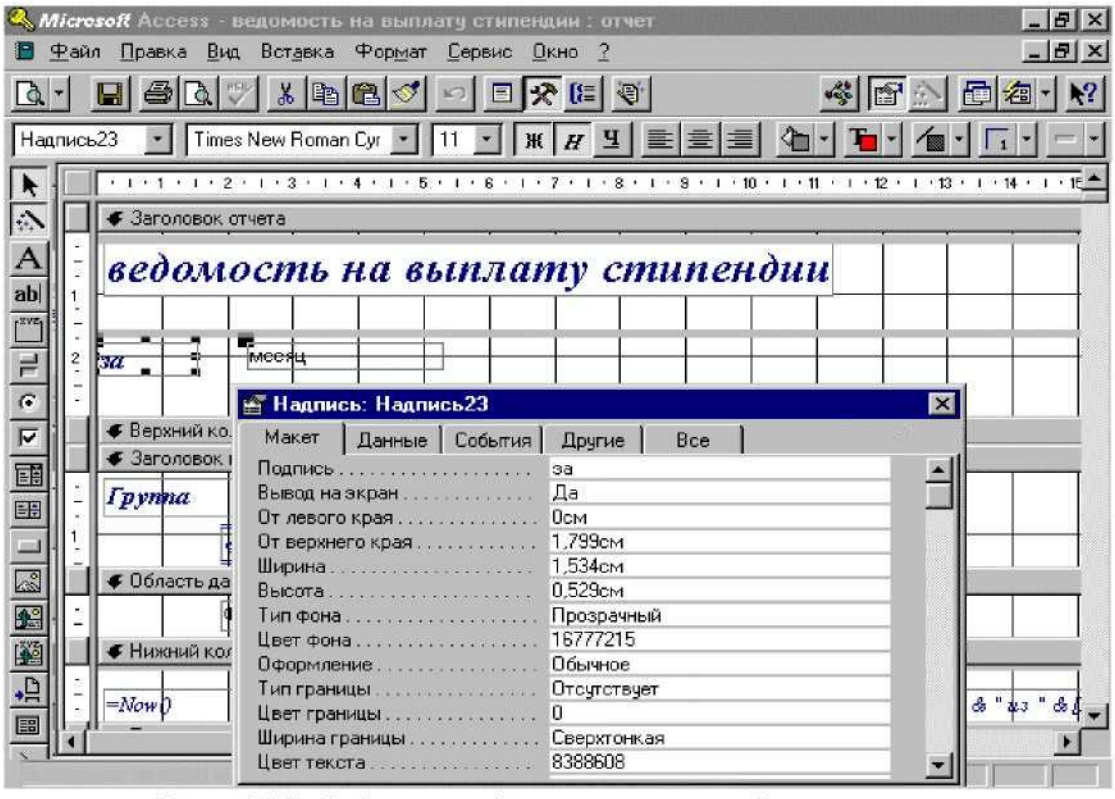


Рис. 2.115. Задание подписи поля в свойствах элемента

Тогда при открытии отчета появиться окно для ввода параметра (рис. 2.116). После ввода в него требуемого значения, оно будет выводиться в отчете (рис. 2.117).

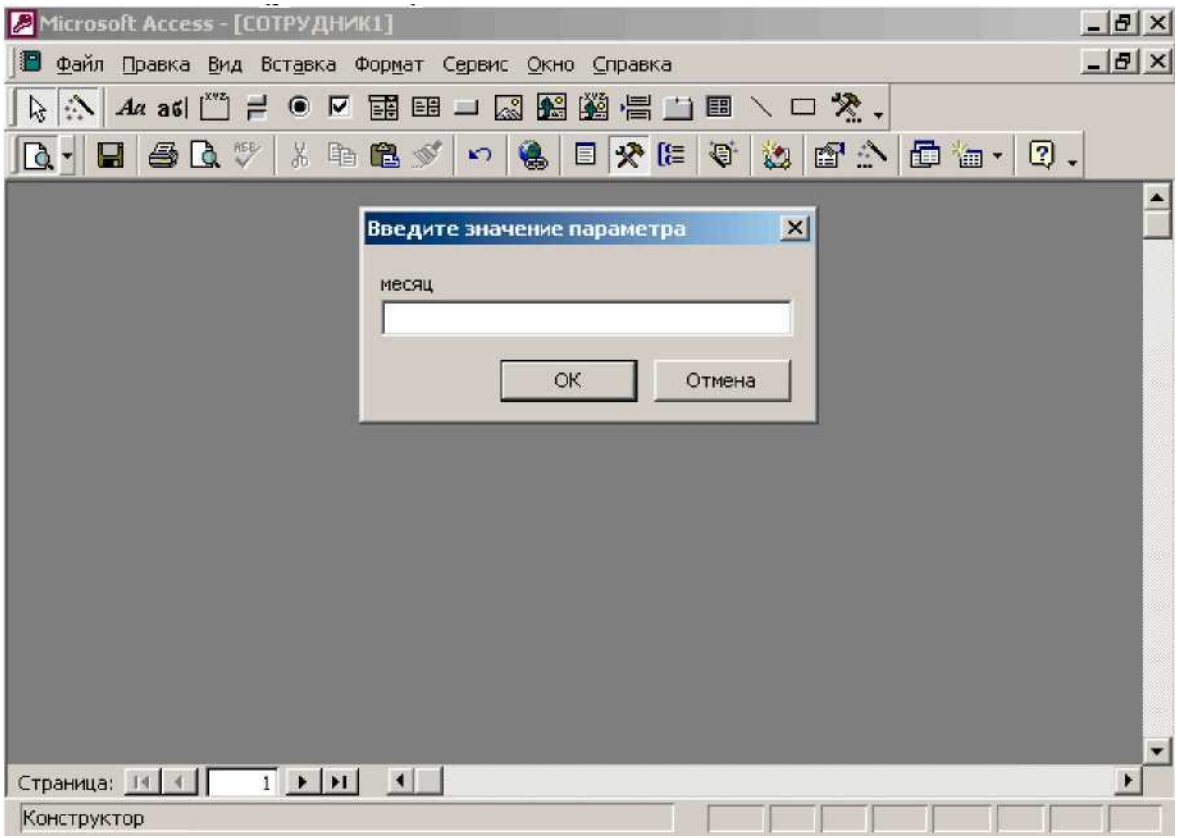


Рис. 2.116. Отчет с параметрами. Запрос на ввод значения параметра

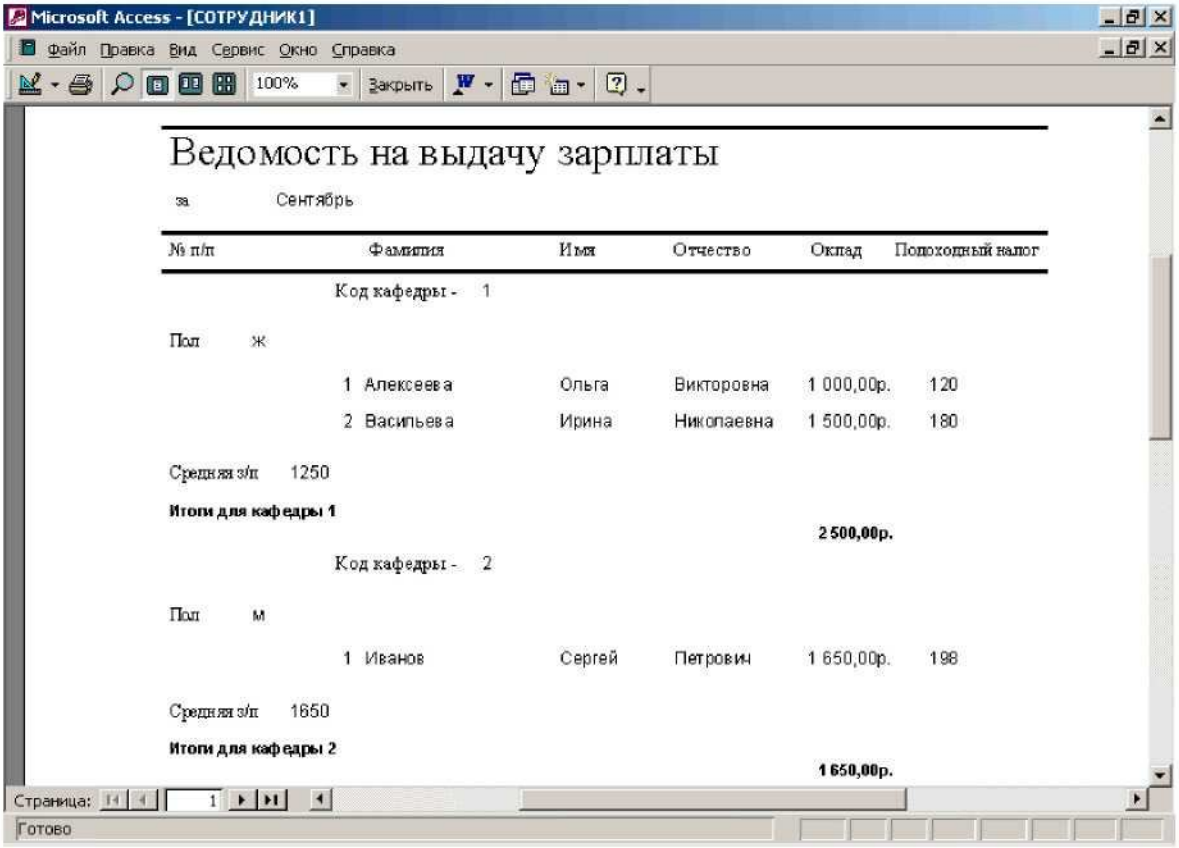


Рис. 2.117. Вывод введенного значения параметра в отчете

Аналогично следует ввести параметр «год».

Следует обратить внимание на то, что назначение «параметра» в параметрическом запросе и параметрическом отчете, отличаются друг от друга: в параметрическом запросе по значению параметра отбираются данные, а в параметрическом отчете это значение просто отображается в том виде, в котором оно было введено.

Определение конца страницы

Обычно ведомость на выплату зарплаты формируют по каждой кафедре отдельно. Чтобы достичь этого, надо перенести информацию из заголовка отчета в заголовок группы, область заголовка отчета вообще закрыть, а вконце области примечания группы поставить признак конца страницы (). Полученный отчет в режиме конструктора будет иметь вид как на рис. 2.118, а в предварительном режиме просмотра - рис. 2.119.

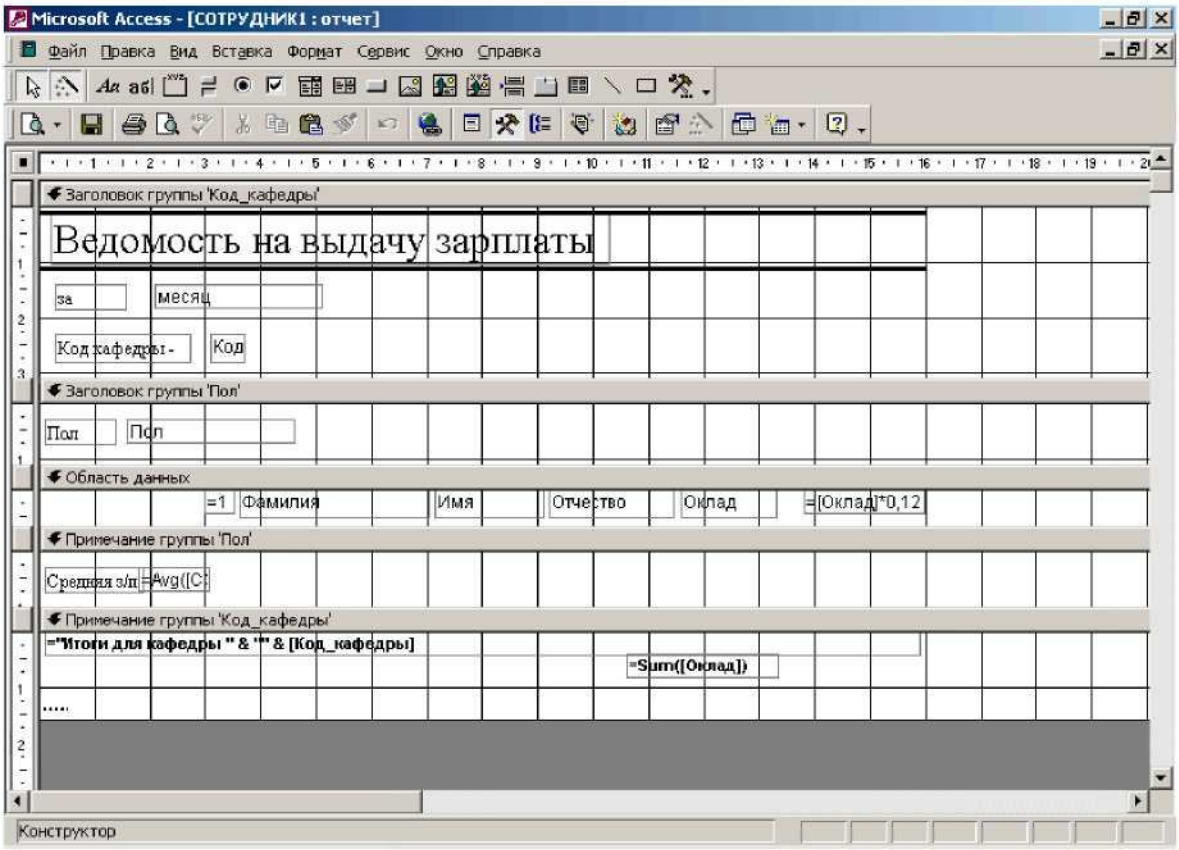


Рис. 2.118. Ведомость на выдачу зарплаты в режиме конструктора

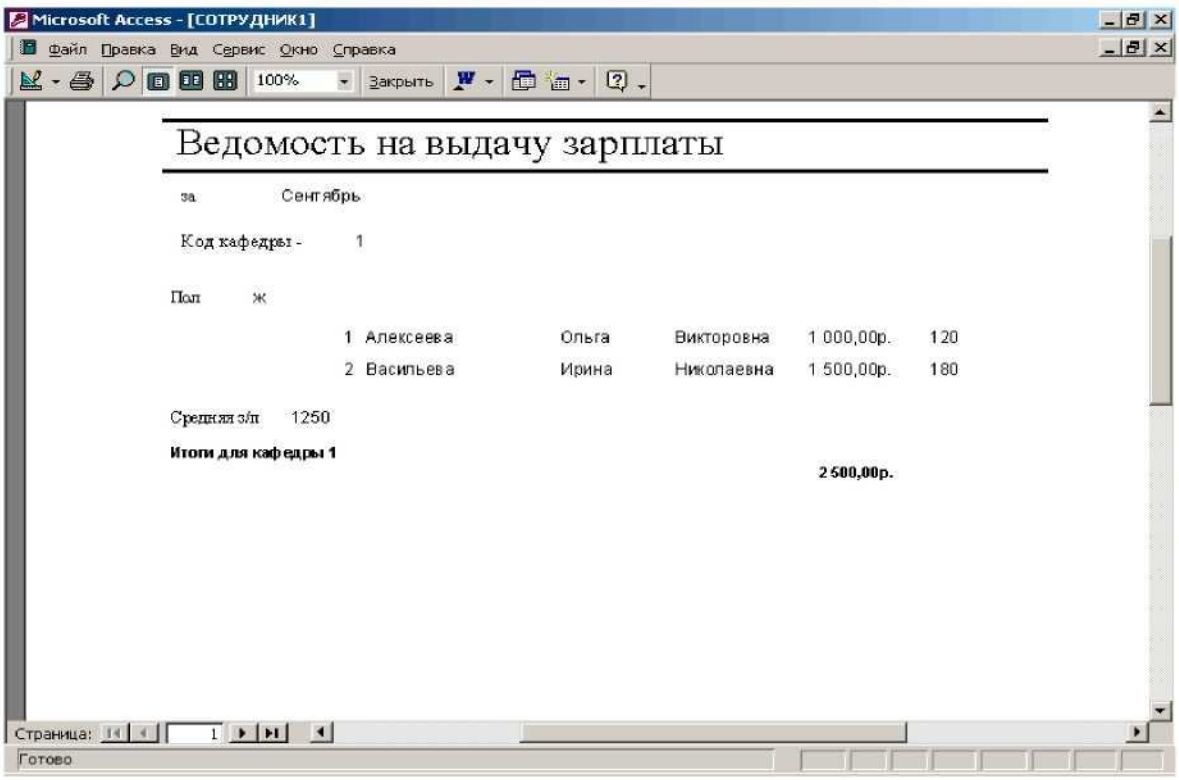


Рис. 2.119. Ведомость на выдачу зарплаты в режиме предварительного просмотра

Обеспечить печать каждой группы с новой страницы можно и иным способом, а именно: в свойствах примечания группы задать признак конец страницы после группы (рис. 2.120)

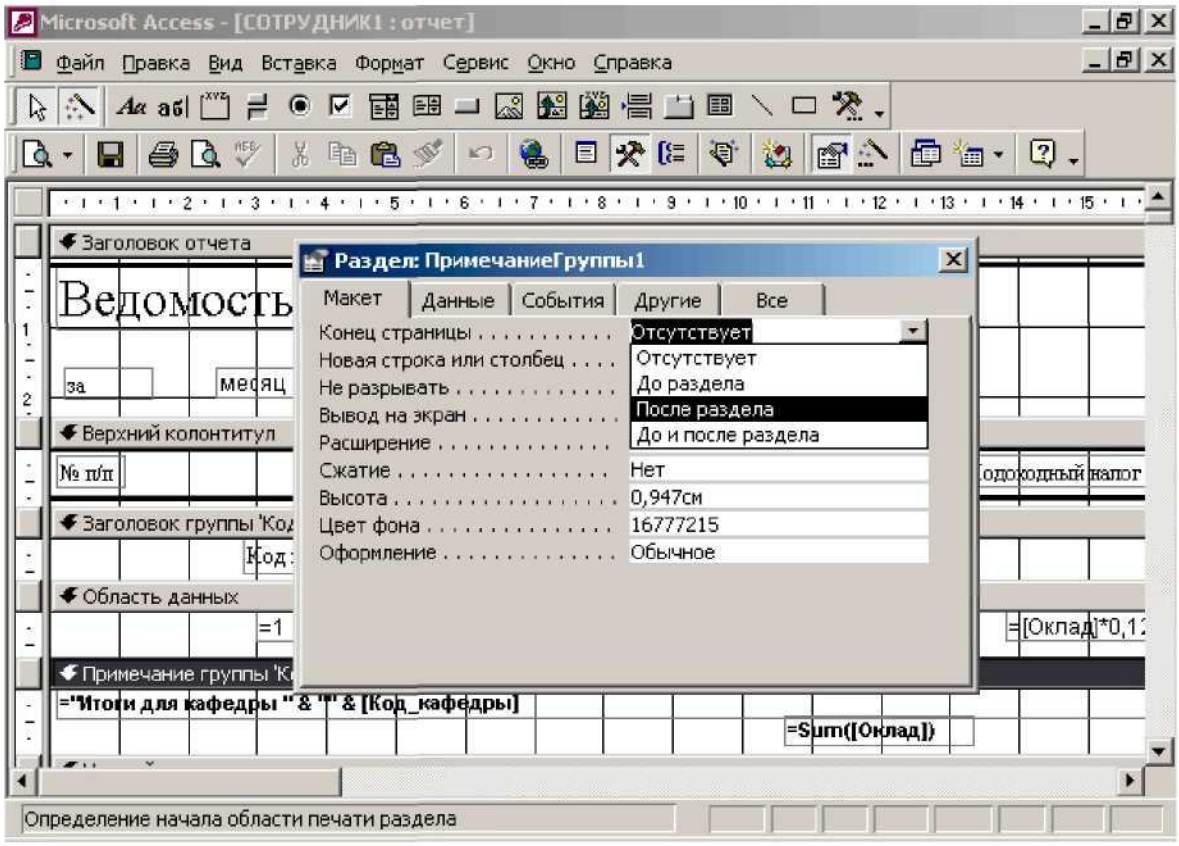


Рис. 2.120. Задание признака конца страницы после группы

2.4.3. Разновидности отчетов

Кроме отчетов табличной формы, о которых шла речь выше, в практике используются и другие формы отчетов, и Access позволяет легко их создавать.

Создание отчета анкетной формы

Для создания отчета анкетной формы (рис. 2.122) надо:

1. В окне базы данных выбрать вкладку **«Отчеты».**
2. Нажать кнопку **«Создать».**
3. **В** диалоговом окне **«Новый отчет»** выбрать **«Автоотчет: в столбец»** (рис. 2.121); в этом случае каждое поле образует отдельную строку с заголовком слева.

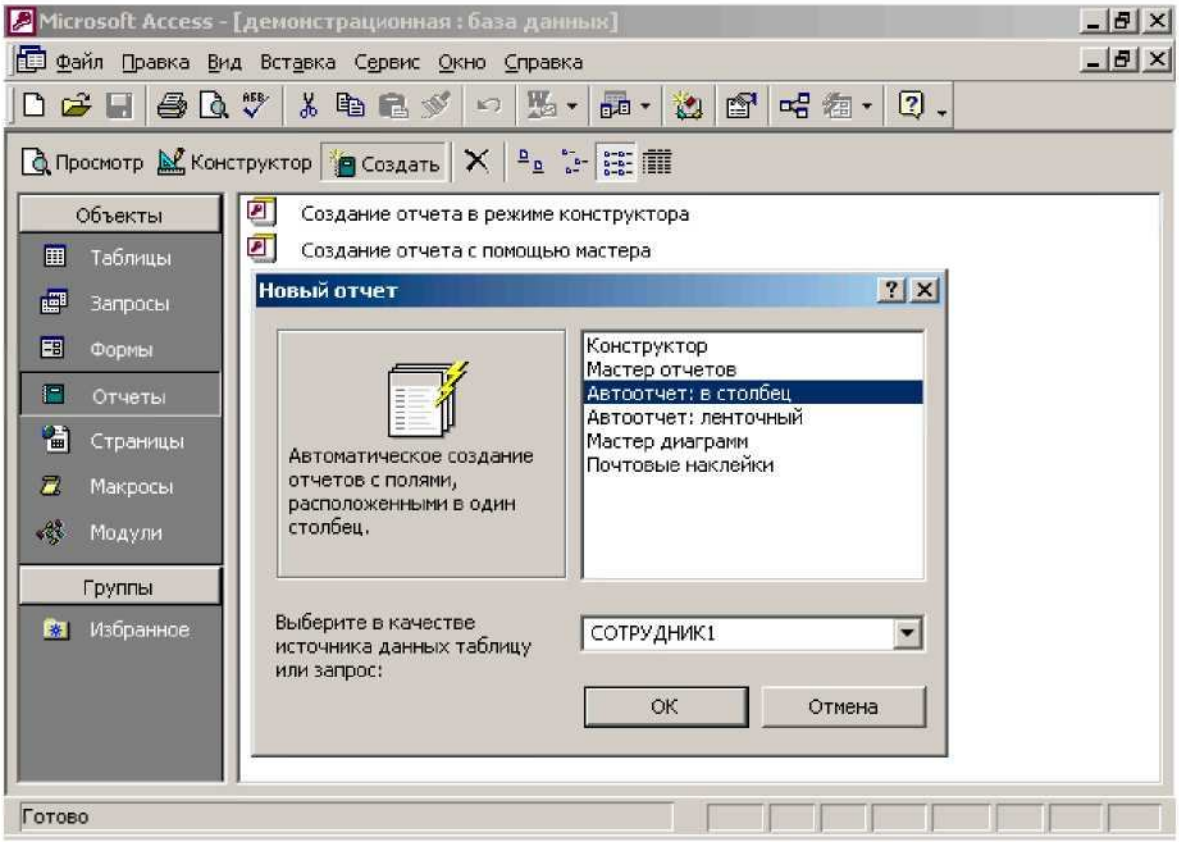


Рис. 2.121. Создание отчета анкетной формы

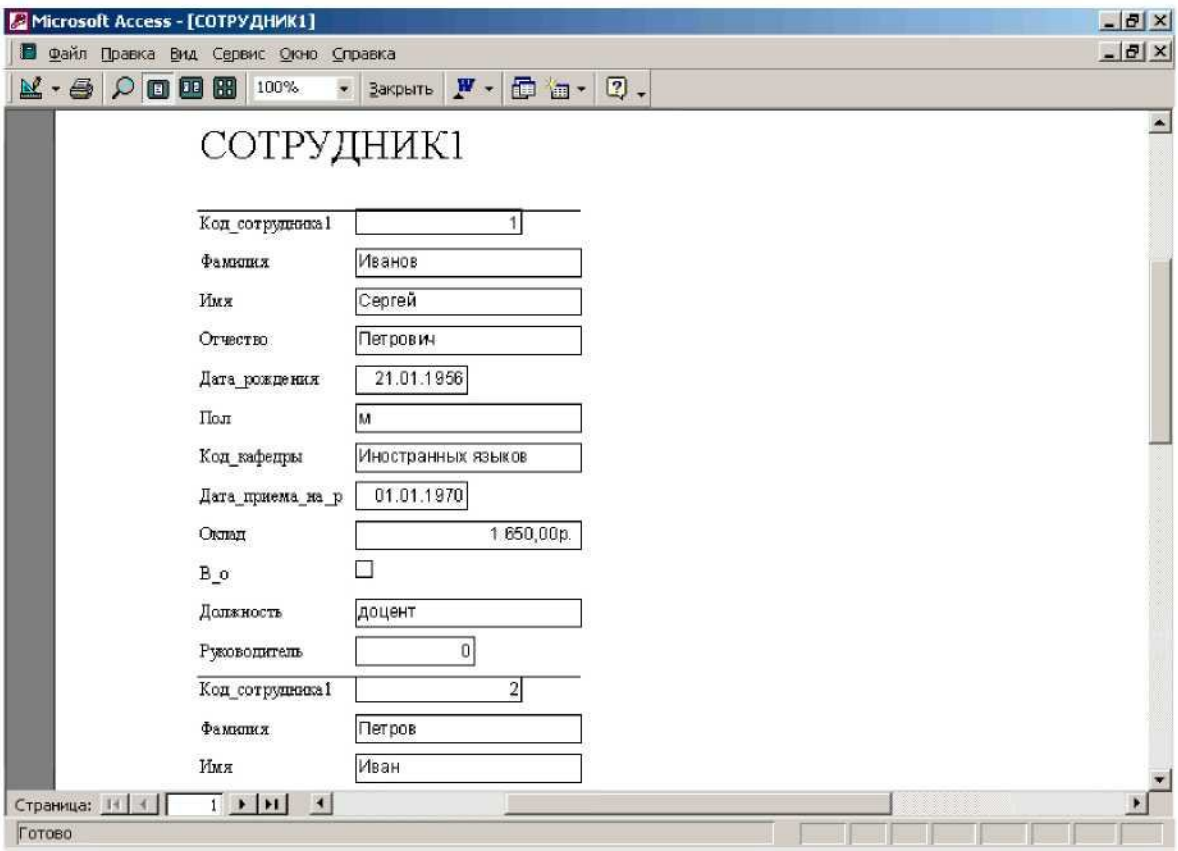


Рис. 2.122. Фрагмент документа анкетной формы

После того, как отчет приобретет нужный вид, его следует сохранить. Для этого можно либо выполнить команду «Файл-Сохранить как», либо попытаться закрыть отчет и на вопрос о том, следует ли сохранять сделанные изменения ответить утвердительно, после чего дать отчету имя.

Создание отчетов в виде «этикеток»

Отчеты формы «этикеток» называются также «почтовыми наклейками» или в английской нотации «Label». Такого типа отчеты обычно включают небольшие число полей, имеют компактный размер (на одном листе размещается несколько экземпляров этикеток обычно размещаемых в несколько колонок). Такие отчеты могут использоваться для создания ценников, беджей и т. п. Для создания таких документов может использоваться специальный мастер по созданию почтовых наклеек.

Предположим, что надо отпечатать беджи для каждого из преподавателей. «Этикетка» будет включать поля ФИО и ДОЛЖНОСТЬ, содержащиеся в таблице СОТРУДНИК, и НАЗВА-НИЕ_КАФЕДРЫ_КРАТКОЕ, содержащиеся в таблице КАФЕДРЫ. Создадим предварительно запрос (рис. 2.123) для отбора именно этой информации и используем его в качестве источника для формирования создаваемого отчета.

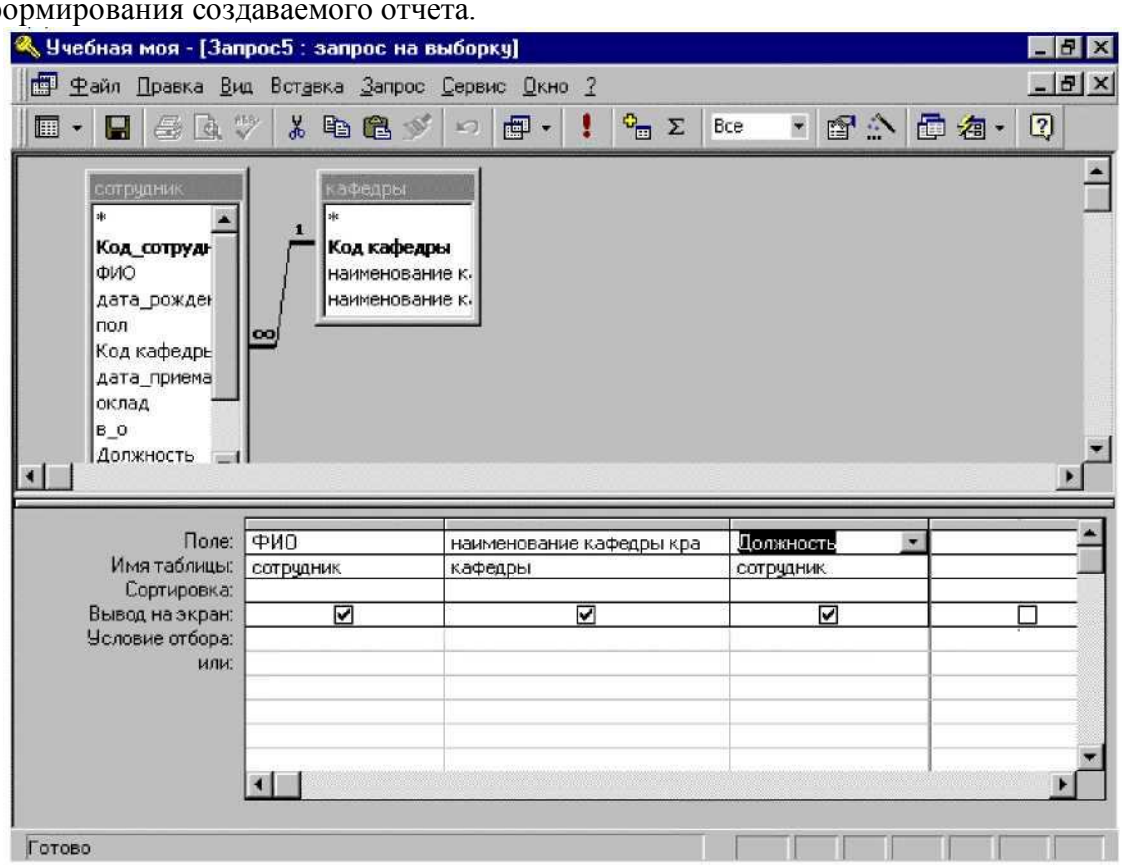


Рис. 2.123. Создание запроса. Подготовительный шаг.

После этого создадим новый отчет, выбрав мастер «почтовых наклеек» (рис. 2.124).

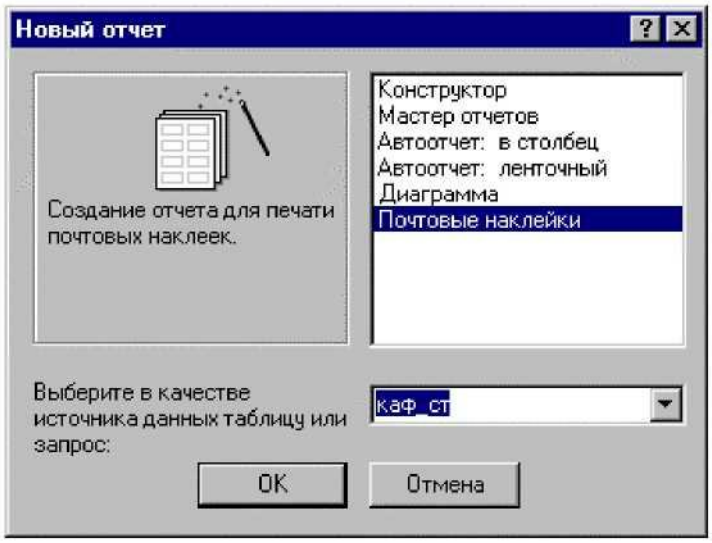


Рис. 2.124. Создание отчетов типа «этикета». Шаг 1.

На следующем шаге можно выбрать либо какой-то подходящий размер этикетки из списка стандартных этикеток, либо задать нужный размер самостоятельно (что мы и сделаем; отметив галочкой параметр «наклейки других размеров» (рис. 2.125).

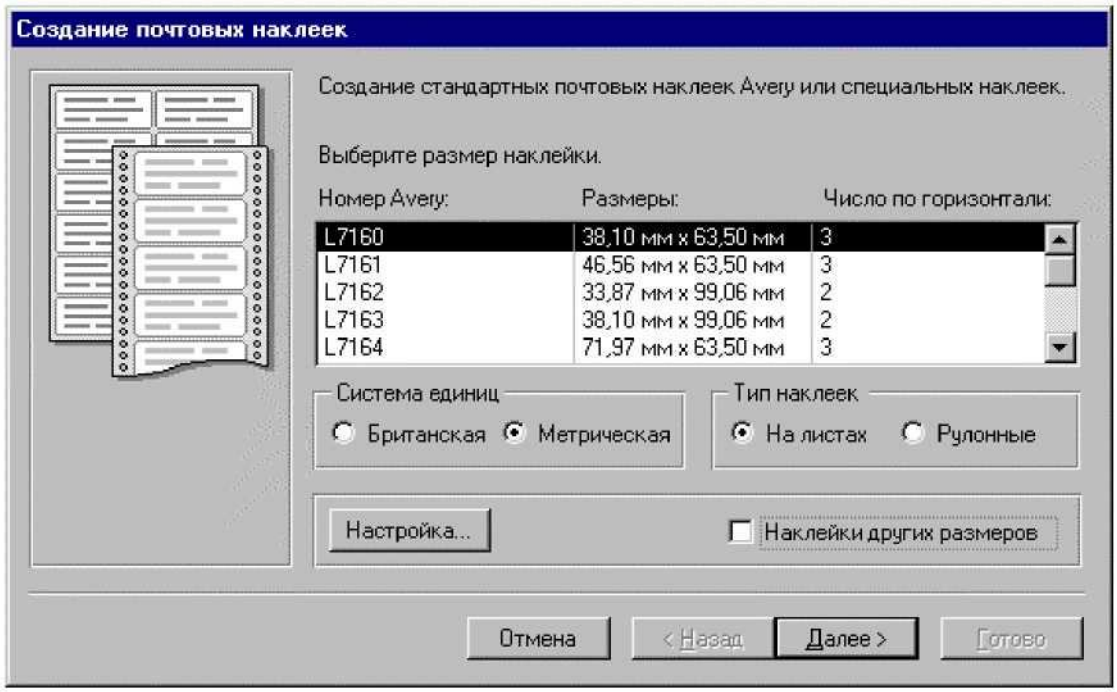


Рис. 2.125. Создание отчетов типа «этикета». Шаг 2.

Далее можно задать размер самой этикетки, поля, расстояние между этикетками (рис. 2.126).

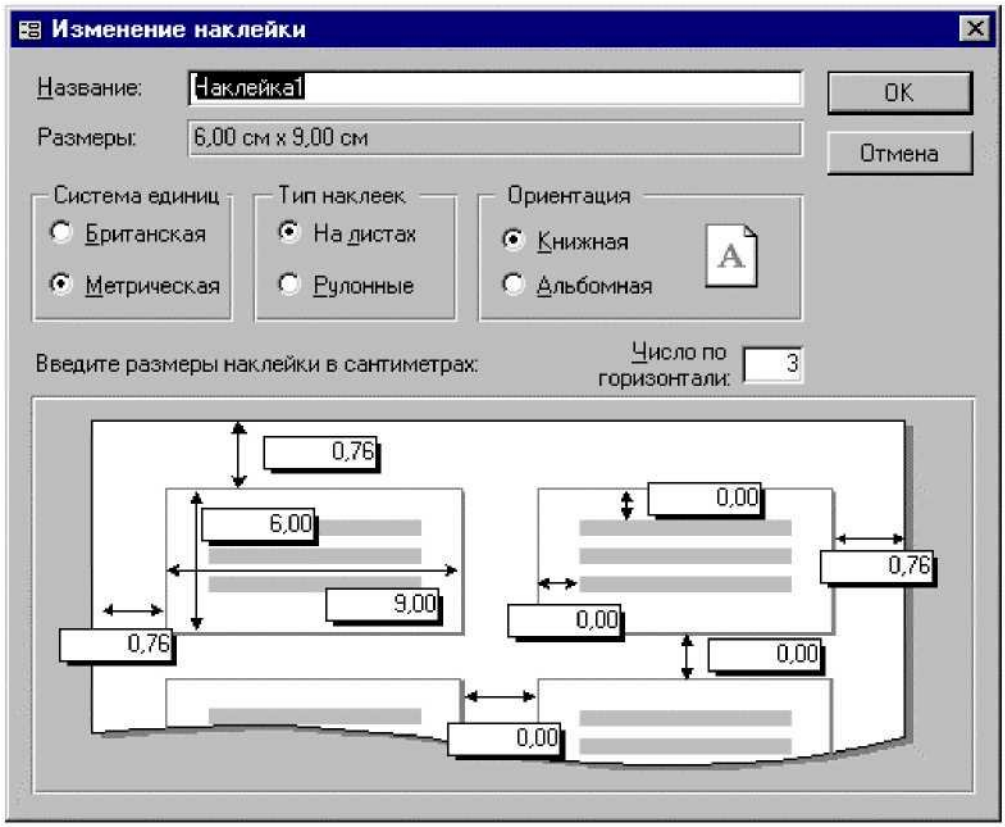


Рис. 2.126. Создание отчетов типа «этикета». Шаг 3.

Далее выбирается шрифт, цвет и некоторые другие оформительские характеристики. После чего делается прототип наклейки (рис. 2.127).

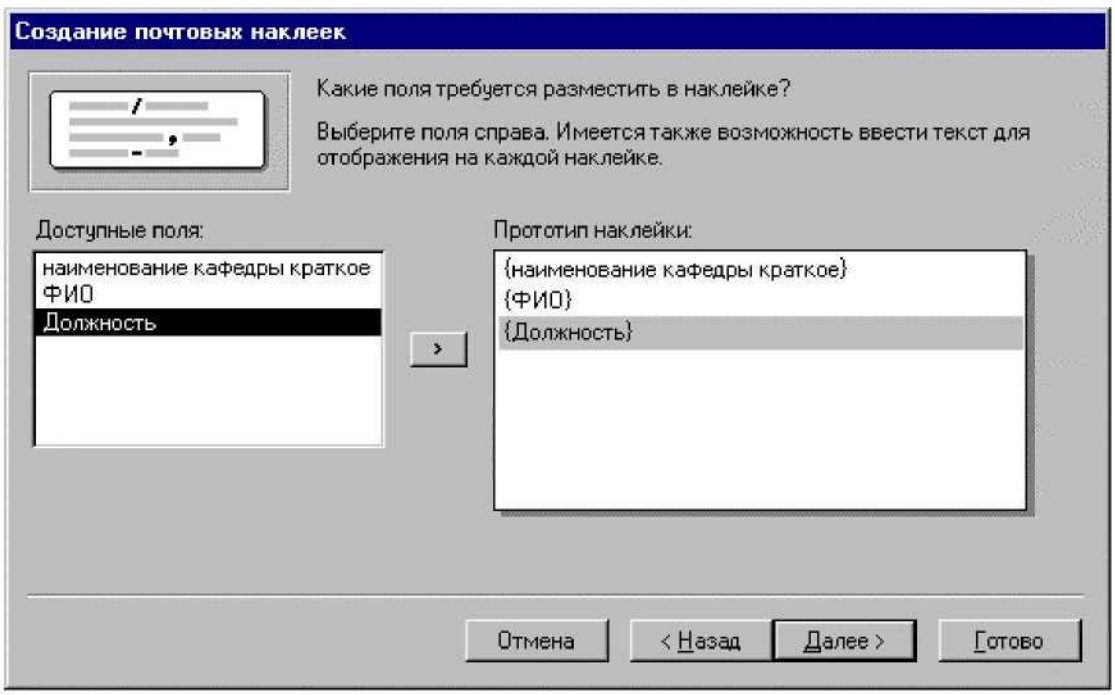


Рис. 2.127. Создание отчетов типа «этикета». Шаг 4.

Если есть необходимость, то наклейки можно отсортировать. Как и любой другой отчет, рассматриваемый тип отчета можно создавать и без помощи мастера.

2.4.4. Совместная работа с другими приложениями MS Office

Существуют несколько способов использования данных Microsoft Access в Microsoft Word. Число возможностей зависит от версии используемого программного продукта.

Для Связи с Office предназначена специальная кнопка L^l

В контексте данного раздела учебника наибольший интерес представляют возможности получения сложных документов, а не просто операции экспорта/импорта.

Если позиционироваться на какую-либо таблицу и нажать на стрелочку на кнопке «Связи с Office», то высветится список возможных вариантов взаимодействия:

- слияние с MS Word
- публикация в MS Word
- анализ в MS Excel.

Если выбрать вариант «слияние с MS Word», то можно создавать документы Word, в которые включаются значения полей из выбранной таблицы. После этого выбора этого варианта откроется соответствующее окно (рис. 2.128).

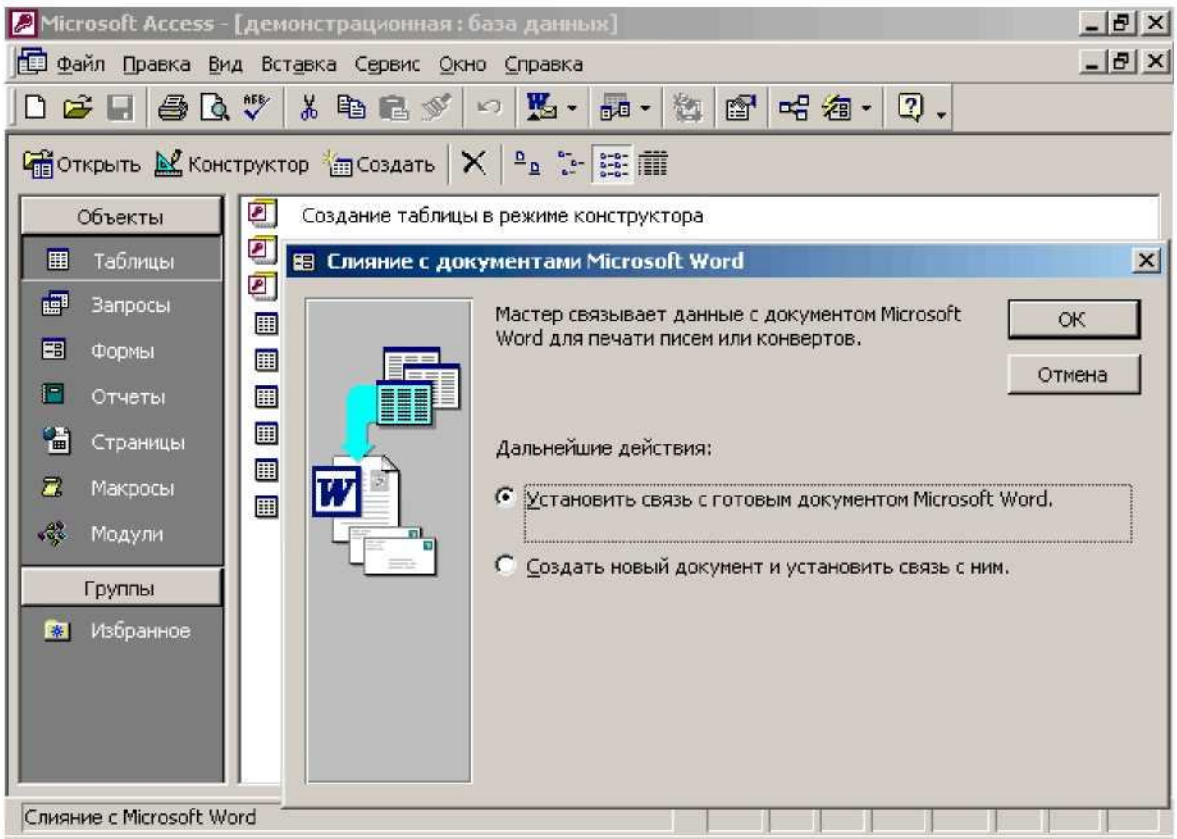


Рис. 2.128. Слияние с документами Microsoft Word

Предположим, что мы хотим напечатать поздравительные открытки своим сотрудникам. Текст у нас заранее не подготовлен, поэтому выберем вторую из предлагаемых альтернатив (создание нового документа и установление связи с ним). После этого открывается документ Word. Текст, который без изменения будет печататься в каждом документе, набирается как обычно. В том месте документа, куда надо вставить значения поля из таблицы/запроса, следует нажать на стрелку на кнопке на панели **«Слияние»,** которая называется «Добавить поле слияния». В открывшемся списке следует выбрать то поле, которое вы хотите вставить в документ. В режиме создания документа в этом месте появиться ссылка на выбранное поле.

Мы в нашем поздравлении хотим использовать обращение «дорогой/дорогая», которое зависит от пола сотрудника. Для того чтобы выбиралось нужное обращение, воспользуемся кнопкой «Добавить поле Word» на панели инструментов **Слияние** и в появившемся списке выберем строку «IF… THEN…ELSE». Поле IF обозначает выполнение одной из двух операций, в зависимости от выполнения указанных условий.

Подготовленный нами текст будет иметь вид:

Дорогая «ФИО» «ФИО»!
Поздравляем Вас с наступающим Новым Годом!
Счастья! Здоровья! Успехов в нашей совместной деятельности!

Вместо ссылок при просмотре и печати будут выводиться конкретные имена и отчества сотрудников.

Кроме получения документов слиянием можно, установив связь с MS Word, воспользоваться позицией меню **«Сервис»-«Конверты и наклейки».**

Для получения сложных документов можно также возможностью **«Публикация в MS Word».** Публиковать можно таблицы, запросы, отчеты.

Генераторы отчетов современных СУБД обладают широкими возможностями по созданию документов различной формы. Мы не касались здесь вопросов форматирования, вставки в отчет деловой графики, которые являются стандартными для многих офисных приложений. Включение в состав СУБД средств, позволяющих представлять информацию в различном виде, удобном не только для ее отображения, но и для анализа, придает этим системам признаки OLAP-систем.

Вопросы для повторения

1. Что в Access называется базой данных?
2. К какому классу относится СУБД Access?
3. Каковы особенности реляционной модели данных?
4. Как создать новую базу данных в Access?
5. Как добавить новый объект в существующую базу данных?
6. Назовите способы создания таблиц?
7. Какие типы полей допустимы в Access?
8. Перечислите способы создания полей подстановки?
9. Какие преимущества дает использование полей подстановки?
10. Какие ограничения накладываются на имена полей?
11. Что называется ключом таблицы?
12. Какими способами можно создать ключ?
13. Является ли наличие ключа в таблице Access обязательным?
14. В каких случаях задание ключа является обязательным?
15. Какими специфическими особенностями обладает поле типа «счетчик»?
16. Назовите свойства полей?
17. Как можно изменить структуру существующей таблицы?
18. Как можно задать объединение таблиц?
19. Что такое «ограничения целостности»?
20. Перечислите виды ограничений целостности?
21. В чем важность задания ограничений целостности?
22. Что такое «ограничение целостности связи» и как они могут задаваться в Access?
23. Назовите способы задания ограничений целостности в Access?
24. Каким образом можно создавать запросы на языке QBE в Access?
25. Какие еще языки запросов можно использовать в Access?
26. Что может служить источником данных для запроса?
27. Перечислите разновидности запросов?
28. Какие запросы называются «сложными»?
29. Как задаются условия «И» и «ИЛИ» в запросах?
30. В чем особенности выполнения запросов на связанных таблицах?
31. Что собой представляют перекрестные запросы?
32. Что собой представляют параметрические запросы?
33. Как вводятся в запрос вычисляемые поля?
34. Как можно получать итоговые значения в запросах?
35. Перечислите разновидности корректирующих запросов?
36. Что значит «открыть запрос»?
37. Что происходит при открытии корректирующего запроса?
38. Как можно сохранить запрос?
39. Как можно сохранить результат выполнения запроса?
40. Как можно задать диапазон в условии запроса?
41. Как задается состав полей, выводимых в ответ?
42. Как можно упорядочить данные в ответе?

43. Каково назначение экранных форм?
44. В каких режимах можно работать с экранной формой?
45. Какими способами можно создавать экранную форму?
46. Как можно включать поля таблицы/запроса в форму при создании формы с помощью «Мастера»?
47. Как можно включать поля таблицы/запроса в форму при работе в режиме конструктора?
48. Как можно скорректировать ранее созданную экранную форму?
49. Какие элементы управления могут быть использованы в экранной форме?
50. Каким образом можно менять размещение элементов на экране?
    Как можно менять размер элемента управления?
51. Чем отличается элемент типа «Список» от «Поле со списком»?
52. Как можно преобразовывать один тип элемента в другой?
53. Какие разновидности многостраничных форм можно создавать в Access?
54. Что такое «многотабличные» формы?
55. Что может являться источником данных для экранной формы?
56. Как можно включить в отчет вычисляемое поле?
57. Как можно включить в отчет рисунок?
58. Как можно создать форму для ввода данных?
59. Каково назначение отчетов?
60. Назовите разновидности отчетов?
61. Какие области выделяются в отчете?
62. Как можно открыть и закрыть ту или иную область?
63. Как вводятся в отчеты вычисляемые поля?
64. Что может являться источником данных для отчетов?
65. Что такое «параметрический отчет»?
66. Как можно скорректировать существующий отчет?
67. В каких режимах можно работать с отчетом?
68. Как можно включать поля таблицы/запроса в отчет при работе в режиме конструктора?
69. Какие элементы управления могут быть использованы в отчете форме?
70. Каким образом можно менять размещение элементов в отчете?
71. Как можно менять размер элемента управления?
72. Каковы особенности использования отчета в качестве источника для которого используется запрос со «*»?
73. Как можно включить в отчет рисунок?
74. Как можно задавать группировку данных в отчете?
75. Как можно сортировать данные в отчете?
76. Как можно разлиновать строки в многострочной части документа?
77. Как можно обеспечить нумерацию строк в отчете?
78. Что такое сложные отчеты и как их можно создавать?
79. Как можно осуществить слияние БД с документами Word?

Резюме по теме

Создание базы данных начинается с создания таблиц, в которых и хранится информация о предметной области. База данных обычно включает несколько взаимосвязанных таблиц. Каждая реляционная таблица по определению имеет ключ. Access позволяет задавать ключ при описании таблицы, но также разрешает и отказаться от этой возможности. По ключу система автоматически выполняет индексирование, а также проверяет уникальность значений ключа при вводе новых записей или их корректировке. Обеспечение целостности БД является одной из важнейших задач при создании БД, так как обеспечение адекватности базы данных отображаемой предметной области является одним из основных требований, предъявляемых к БД. После описания таблицы можно сразу вводить в нее данные. Но такой способ имеет многие очевидные недостатки. Поэтому для этих целей обычно используются так называемые экранные формы. После описания таблиц и заполнения их данными к базе данных можно формулировать разнообразные запросы. Создание отчетов является важной функцией, предоставляемой СУБД, так как именно отчеты позволяют представить данные из баз данных в удобном виде.

Тема 3. *Введение в интеллектуальные системы*

Цели и задачи изучения темы

Целью изучения темы является формирование устойчивого представления об основах искусственного интеллекта и основанных на нем способах представления знаний.

Задачи изучения темы:

- изучить современное состояние исследований в области искусственного интеллекта и их основные направления;
- изучить способы представления знаний и организации вывода на знаниях;
- рассмотреть основы нечетких знаний и нечеткой логики;
- рассмотрение прикладных интеллектуальных систем.

3.1. Основные направления исследований в области искусственного интеллекта (ИИ)

Синтезируя десятки определений ИИ из различных источников, изданной книге в качестве рабочего определения можно предложить следующее.

Искусственный интеллект — это одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными задачи, общаясь с ЭВМ на ограниченном подмножестве естественного языка.

Среди множества направлений искусственного интеллекта есть несколько ведущих, которые в настоящее время вызывают наибольший интерес у исследователей и практиков. Опишем их чуть подробнее.

Представление знаний и разработка систем, основанных на знаниях (knowledge-based systems). Это основное направление в области изучения искусственного интеллекта. Оно связано с разработкой моделей представления знаний, созданием баз знаний, образующих ядро экспертных систем. В последнее время включает в себя модели и методы извлечения и структурирования знаний и сливается с инженерией знаний. Именно исследованиям в этой области посвящена данная книга. Подробнее смотрите главы 2-5.

Программное обеспечение систем ИИ (software engineering for Al). В рамках этого направления разрабатываются специальные языки для решения интеллектуальных задач, в которых традиционно упор делается на преобладание логической и символьной обработки над вычислительными процедурами. Эти языки ориентированы на символьную обработку информации – LISP, PROLOG, SMALLTALK, РЕФАЛ и др. Помимо этого создаются пакеты прикладных программ, ориентированные на промышленную разработку интеллектуальных систем, или программные инструментарии искусственного интеллекта, например, KEE, ARTS, G2 [Хейес-Рот и др., 19S7; Попов, Фоминых, Кисель, Шапот, 1996] Достаточно популярно также создание так называемых пустых экспертных систем или «оболочек» - КАРРД, EXSYS, Ml, ЭКО и др. базы знаний которых можно наполнять конкретными знаниями, создавая различные прикладные системы. Подробно эти технологии рассмотрены в главе 6.

Разработка естественно-языковых интерфейсов и машинный перевод (natural language processing). Начиная с 50-х годов одной из популярных тем исследований в области ИИ является компьютерная лингвистика, и, в частности, машинный перевод (МП). Идея машинного перевода оказалась совсем не так проста, как казалось первым исследователям и разработчикам.

Уже первая программа в области естественно-языковых (ЕЯ) интерфейсов – переводчик с английского на русский язык — продемонстрировала неэффективность

первоначального подхода, основанного на пословном переводе. Однако, еще долго разработчики пытались создать программы на основе морфологического анализа. Неплодотворность такого подхода связана с очевидным фактом: человек может перевести текст только на основе пониманий его смысла и в контексте предшествующей информации, или контекста. Иначе появляются переводы в стиле «Моя дорогая Маша — my expensive Masha». В дальнейшем системы МП усложнялись, и в настоящее время используется несколько более сложных моделей:

• применение так называемых «*языков-посредников»* или языков смысла, в результате происходит дополнительная трансляция «исходный язык оригинала — язык смысла — язык перевода»;

• *ассоциативный поиск* аналогичных фрагментов текста и их переводов в специальных текстовых репозиториях или базах данных;

• *структурный подход,* включающий последовательный анализ и синтез естественно-языковых сообщений. Традиционно такой подход предполагает наличие нескольких фаз анализа:

1. Морфологический анализ — анализ слов в тексте.
2. Синтаксический анализ — разбор состава предложений и грамматических связей между словами.

3, Семантический анализ — анализ смысла составных частей каждого предложения на основе некоторой предметно-ориентированной базы знании.

4. Прагматический анализ — анализ смысла предложений в реальном контексте на основе собственной базы знаний.

Синтез ЕЯ-сообщений включает аналогичные этапы, но несколько в другом порядке. Подробнее смотрите работы [Попов, 1982; Мальковский, 1985]

Интеллектуальные роботы (robotics). Идея создания роботов далеко не нова. Само слово «робот» появилось в 20-х годах, как производное от чешского «робота» — тяжелой грязной работы. Его автор — чешский писатель Карел Чапек, описавший роботов в своем рассказе.Роботы — это автоматические устройства, предназначенные для автоматизации человеческого труда.

Можно условно выделить несколько поколений в истории создания и развития робототехники:

I поколение. Роботы с жесткой схемой управления. Практически все современные промышленные роботы принадлежат к первому поколению. Фактически это программируемые манипуляторы.

II поколение. Адаптивные роботы с сенсорными устройствами. Есть образцы таких роботов, но в промышленности они пока используются мало.

III поколение. Самоорганизующиеся или интеллектуальные роботы. Это — конечная цель развитии робототехники. Основные нерешенные проблемы при создании интеллектуальных роботов — проблема машинного зрения и адекватного хранения и обработки трехмерной визуальной информации.

В настоящее время в мире изготавливается более 60 000 роботов в год. Фактически робототехника сегодня — это инженерная наука, не отвергающая технологий ИИ, но не готовая пока к их внедрению в силу различных причин.

Обучение и самообучение (machine learning). Активно развивающаяся область искусственного интеллекта включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний на основе анализа и обобщения данных [Гаек, Гавранек, 1983; Гладун, 1994; Финн, 1991]. Включает обучение по примерам (или индуктивное), а также традиционные подходы из теории распознаваний образов.

В последние годы к этому направлению тесно примыкают стремительно развивающиеся системы data mining — анализа данных и knowledge discovery — поиска закономерностей в базах данных.

Распознавание образов (pattern recognition). Традиционно — одно из направлений искусственного интеллекта, берущее начало у самых его истоков, но в настоящее время практически отделившееся в самостоятельную науку. Ее основной подход — описание классов объектов через определенные значения значимых признаков. Каждому объекту ставится в соответствие матрица признаков, но которой происходит его распознавание. Процедура распознавания использует чаще всего специальные математические процедуры и функции, разделяющие объекты на классы. Это направление близко к машинному обучению и тесно связано с нейрокибернетикой [Справочник поИИ1990].

Новые архитектуры компьютеров (new hardware platforms and architectures).Самые современные процессоры сегодня основаны на традиционной последовательной архитектуре Фон Неймана, используемой еще в компьютерах первых поколений. Эта архитектура крайне неэффективна для символьной обработки. Поэтому усилия многих научных коллективов и фирм уже десятки лет нацелены на разработку аппаратных архитектур, предназначенных для обработки символьных и логических данных. Создаются Пролог - и Лисп-машины, компьютеры V и VI поколений. Последние разработки посвящены компьютерам баз данных параллельным и векторным компьютерам [Амамия, Танака, 1993].

И хотя удачные промышленные решения существуют высокая стоимость, недостаточное программное оснащение и аппаратная несовместимость с традиционными компьютерами существенно тормозят широкое использование новых архитектур.

8. Игры и машинное творчество. Это, ставшее скорее историческим, направление связано с тем, что на заре исследований ИИ традиционно включал в себя игровые интеллектуальные задачи – шахматы, шашки. В основе первых программ лежит один из ранних подходов — лабиринтная модель мышления и эвристики. Сейчас это скорее коммерческое направление, так как в научном плане эти идеи считаются тупиковыми.

Кроме того, это направление охватывает сочинение компьютером музыки [Зари-пов1 1983], стихов, сказок [Справочник по ИИ, 1986] и даже афоризмов [Любич, 1998]. Основным методом подобного «творчества» является метод пермутации (перестановок) плюс использование некоторых баз знаний и данных, содержащих результаты исследований по структурам текстов, рифм, сценариям и т. и.

9. Другие направления. ИИ – междисциплинарная наука, которая, как мощная река по дороге к морю, вбирает в себя ручейки и речки смежных наук. Выше перечислены лишь те направления, которые прямо или косвенно связаны с основной тематикой учебника — инженерией знаний. Стоит лишь взглянуть на основные рубрикаторы конференций по ИИ, чтобы понять, насколько широко простирается область исследований по ИИ:

генетические алгоритмы;
когнитивное моделирование;
интеллектуальные интерфейсы;
распознавание и синтез речи;
дедуктивные модели.

3.2. Представление знаний и вывод на знаниях

3.2.1. Данные и знания

При изучении интеллектуальных систем традиционно возникает вопрос — что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых ЭВМ. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным.

Данные — это отдельные факты, характеризующие объекты, процессы и явлений предметной области, а также их свойства.

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

1. D1 — данные как результат измерений и наблюдений;
2. D2 — данные на материальных носителях информации (таблицы, протоколы, справочники);
3. D3 — модели (структуры) данных в виде диаграмм, графиков, функций;
4. D4 — данные в компьютере на языке описания данных;
5. D5 — базы данных на машинных носителях информации.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Знания — это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

При обработке на ЭВМ знания трансформируются аналогично данным.

1. Z1 — знания в памяти человека как результат мышления;
2. Z2 — материальные носители знаний (учебники, методические пособия);
3. Z3 — *поле знаний* — условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;
4. Z4 — знания, описанные на языках представления знании (продукционные языки, семантические сети, фреймы — см, далее);
5. Z5 — *база знаний на машинных носителях информации.*

Часто используется такое определение знаний: *знания* — это хорошо структурированные данные, или данные о данных, или метаданные.

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. Интенсионал понятия - это определение через понятие более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящиеся к определяемому объекту. Это есть определение через данные, или *экстенсионал понятия.*

Пример 3.1

Понятие «Персональный компьютер». Его интенсионал: «*Персональный компьютер — это дружественная ЭВМ, которую можно поставить на стол и купить менее чем за $2000-3000».*

Экстенсионал этого понятия: «*Персональный компьютер — это Mac, IBM PC, Sinkler».*

Для хранения данных используются базы данных (для них характерны большой объем и относительно небольшая удельная стоимость информации), для хранения знаний — базы знаний небольшого объема, но исключительно дорогие информационные массивы). *База знаний* — основа любой интеллектуальной системы.

Знания могут быть классифицированы по следующим категориям:

— *Поверхностные* — знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области,

— *Глубинные* — абстракции, аналогии схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Пример 3.2

Поверхностные знания: *«Если нажать на кнопку звонка, раздастся звук. Если болит, то следует принять аспирин».*

Глубинные знания: *«Принципиальная электрическая схема звонков и проводки. Знания физиологов и врачей высокой квалификации о причинах, видах болей и методах их лечения».*

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Кроме того, в учебниках по ИИ знания традиционно делят на *процедурные* на *декларативные.* Исторически первичными были процедурные знания, то есть знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы. Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знании сосредоточивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму, то есть знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному и представленных неспециалистам.

3.2.2. Модели представления знаний

Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам:

— продукционные модели;
— семантические сети;
— фреймы;
— формальные логические модели.

Продукционная модель – модель, основанная на правилах, позволяет представить знания в виде предложений типа – «Если (условие), то (действие)».

Под «условием» *(антецедент)* понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под «действием» *(консеквентом)* - действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, уступающими далее как условия и терминальными или целевыми, завершающими работу системы).

Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения - к данным). Данные – это исходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, перебирающий правила из продукционной базы знаний (см. далее).

Продукционная модель чаще всего применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; «оболочки» или «пустые» ЭС — EXSYS Professional, Kappa, ЭКСПЕРТ; ЭКО, инструментальные системы ПИЭС [Хорошевский, 1993] и СПЭИС [Ковригин, Перфильев, 1988] и др.), а также промышленных ЭС на его основе (например, ЭС, созданных средствами G2 [Попов, 1996]) и др.

Продукционная модель чаще всего применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; «оболочки» или «пустые» ЭС — EXSYS Professional, Kappa, ЭКСПЕРТ; ЭКО, инструментальные системы ПИЭС [Хорошевский 1993] и СПЭИС [Ковригин, Перфильев, 1988] и др.), а также промышленных ЭС на его основе (например, ЭС, созданных средствами G2 [Попов, 1996]) и др.

Термин *семантическая* означает «смысловая», а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков. *Сеть* — это ориентированный граф, вершины которого — понятия, а дуги – отношения между ними.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а *отношения* — это связи типа: «это» («АКО — A-Kind-Of» «is»), «имеет частью» («has part»), «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

— класс — элемент класса (цветок — роза);

— свойство — значение (цвет — желтый);

— пример элемента класса (роза — чайная).

Можно предложить несколько классификаций семантических сетей, связанных с тинами отношений между понятиями.

По количеству типов отношений:

— Однородные (с единственным типом отношений).

— Неоднородные – (с различными типами отношений).

По типам отношений:

— Бинарные (в которых отношения связывают два объекта).

— • N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Наиболее часто в семантических сетях используются следующие отношения:

— связи типа *«часть — цело» («класс — подкласс», «элемент — множество»* и т. д.);

— функциональные связи, определяемые обычно глаголами - производит, количественные (больше, меньше, равно);

— пространственные (далеко от, близко от, за, под, над);

— временные (раньше, позже, в течение);

— атрибутивные связи (иметь свойство, иметь значение);

— логические связи (И, ИЛИ, НЕ);

— лингвистические связи и др.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующее по некоторой подсети, отражающей поставленный запрос.

Пример 3.3

На рис. 3.1 изображена семантическая сеть. В качестве вершин тут выступают понятия

«человеку от Иванова», «Волга- автомобили», 4 вида транспорта и «двигатель».

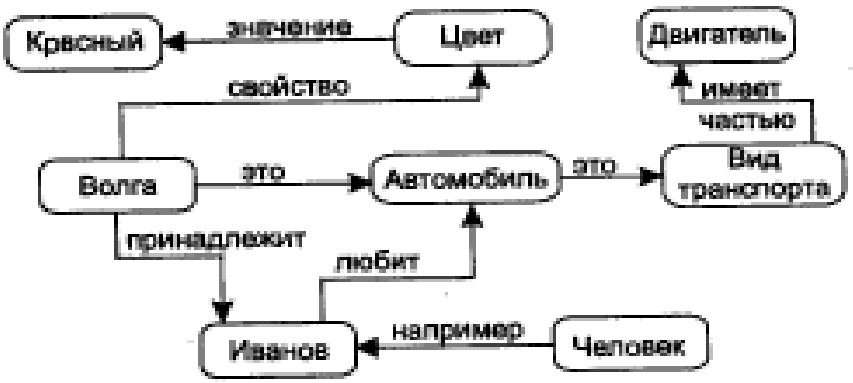


Рис. 3.1. Семантическая сеть

Данная модель представления знаний была предложена американским психологом Киллианом. Основным ее преимуществом является то, что она более других соответствует современным представлениям об организации долговременной памяти человека (Скрэгг, 1983),

Недостатком этой модели является сложность организации процедуры поиска вывода на семантической сети.

Для реализации семантических сетей существуют специальные сетевые языки, например NET [Цейтин, 1985], язык реализации систем SIMER+MIR [Осипов, 1Э97] и др. Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний - PROSPECTOR, CASNET, TORUS [Хейес-Рот и др. 19S7; Durkin, 1998].

Термин *фрейм* (от английскою «frame», что означает «каркас» или «рамка») был предложен *Марвином Ли Минским* [Минский, 1979], одним из пионеров ИИ, в 70-е годы для обозначения структуры знаний для восприятия пространственных сцен. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Фрейм – это абстрактный образ для представления некоторого стереотипа.

В психологи к и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты: «помещение с четырьмя стенами, полом, потолком, окнами и дверь, площадью 6-20 $м^3$». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки» или «слоты» - это незаполненные значения некоторых атрибутов — например, количество окон, цвет стен, высота потолка, покрытие пола и др.

В теории фреймов такой образ комнаты называется фреймом комнаты. Фреймом также называется и формализованная модель для отображения образа

Различают *фреймы-образцы,* или *прототипы,* хранящиеся в базе знаний, и *фреймы-экземпляры,* которые создаются для отображения реальных: фактических ситуаций на основе поступающих данных. Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире через:

— фреймы-*структуры*, использующиеся для обозначения объектов и понятий (заем, залог, вексель);

— фреймы - *роли* (менеджер, кассир, клиент);

— фреймы - *сценарии* (банкротство, собрание акционеров, празднование именин);

— фреймы - ситуации (тревога, авария, рабочий режим устройства) и др.

Традиционно структура фрейма может быть представлена как список свойств:

(ИМЯ ФРЕЙМА;

(имя 1-го слота; значение 1-го слота), (имя 2-го слота: значение 2-го слота),

(имя N-го слота: значение N-го слота)).

Ту же запись можно представить в виде таблицы, дополнив ее двумя столбцами (табл. 3.1).

Таблица 3.1.

Структура фрейма

Имя фрейм			
Имя слота	Значение слота	Способ получения значения	Присоединенная процедура

В таблице дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур, что допускается в теории фреймов. В качестве значения слота может выступать имя другого фрейма, так образуются сети фреймов. Существует несколько способов получения слотом значений во фрейм-экземпляре:

— по умолчанию от фрейма-образца (Default-значение);
— через наследование свойств от фрейма, указанного в слоте АКО;
— по формуле, указанной в слоте;
— через присоединенную процедуру;
— явно из диалога с пользователем;
— из базы данных.

Важнейшим свойством теории фреймов является заимствование из теории семантических сетей — так называемое *наследование свойств.* И во фреймах, и в семантических сетях наследование происходит по АКО-связям (A Kind of = это). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

Пример 3.4.

Например, в сети фреймов на рис. 3.2 понятие «ученик» наследует свойства фреймов «ребенок» и «человек», которые находятся па более высоком уровне иерархии. Так, на вопрос «любят ли ученики сладкое» следует ответ – «да», так как этим свойством обладают все дети, что указано во фрейме «ребенок». Наследование свойств может быть частичным, так как возраст для учеников не наследуется из фрейма «ребенок», поскольку указан явно в своем собственном фрейме.

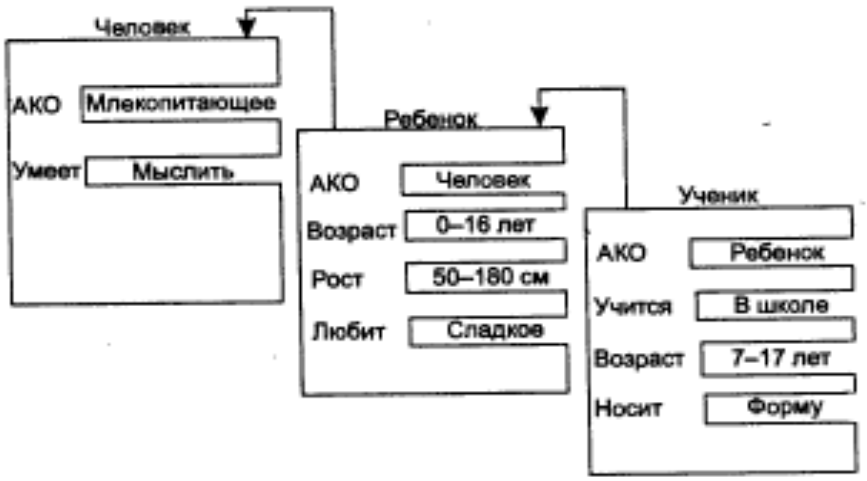


Рис. 3.2. Сеть фреймов

Основные преимущества фреймов как модели представления знаний является способность отражать концептуальную основу организации памяти человека [Шенк, Хантер, 1987], а также ее гибкость и наглядность.

Специальные языки представления знаний в сетях фреймов FRL (Frame Representation Language) [Байдун, Бунин, 1990], KRL (Knowledge Representation Language) [Уотермен, 1989), фреймовая «оболочка» Карра [Стрельников, Бори-сои, 1997] и другие программные средства позволяют эффективно строить промышленные ЭС. Широкоизвестны такие фрейм-ориентированные экспертные системы, как ANALYST, МОДИС, TRISTAN, ALTEftlD [Ковригин, Перфильев, 198S; Николов, 1988; Sisodia, Warbntin, 1992].

Традиционно в представлении знаний выделяют *формальные логические модели, основанные на классическом* исчислении предикатов 1-го порядка, когда предметная область или задача описывается в виде набора аксиом. Мы же опустим описание этих моделей по следующим

причинам. Исчисление предикатов 1-го порядка в промышленных экспертных системах практически не используется. Эта логическая модель применима в основном в исследовательских «игрушечных» системах, так как предъявляет очень высокие требования и ограничения к предметной области.

В промышленных же экспертных системах используются различные ее модификации и расширения, изложение которых выходит за рамки этого учебника.

3.2.3. Вывод на знаниях

Несмотря на все недостатки, наибольшее распространение получила продукционная модель представления знаний. При использовании продукционной модели база знаний состоит из набора правил. *Программа, управляющая перебором правил, называется машиной вывода.*

Машина вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр существующих фактов из рабочей памяти (базы данных) и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов, а во-вторых, определение порядка просмотра и применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных [Осуга, Саэки, 1990].

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой небольшую по объему программу и включает два компонента — один реализует собственно вывод, другой управляет этим процессом. Действие *компонента вывода* основано из применении правила, называемого *modus ponens.*

Правило modus ponens. Если известно, что истинно утверждение А, то существует правило вида -ЕСЛИ А, ТО В», тогда утверждение В также истинно.

Правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение. Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако, система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации. *Управляющий компонент* определяет порядок применения правил и выполняет четыре функции:

1. *Сопоставление* — образец правила сопоставляется с имеющимися фактами,
2. *Выбор* — если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта),
3. *Срабатывание* — если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает.
4. *Действие* — рабочая память подвергается изменению путем добавления в нее заключения сработавшего, правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется (как, например, в системах обеспечения безопасности информации).

Интерпретатор продукций работает циклически. В каждом цикле он просматривает все правила, чтобы выявить те, посылки которых совпадают с известными на данный момент фактами из рабочей памяти, После выбора правило срабатывает, его заключение заносится в рабочую память, и затем цикл повторяется сначала. В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то интерпретатор производит выбор по определенному критерию единственного правила, которое срабатывает в данном цикле. Цикл работы интерпретатора схематически представлен на рис. 3.3.

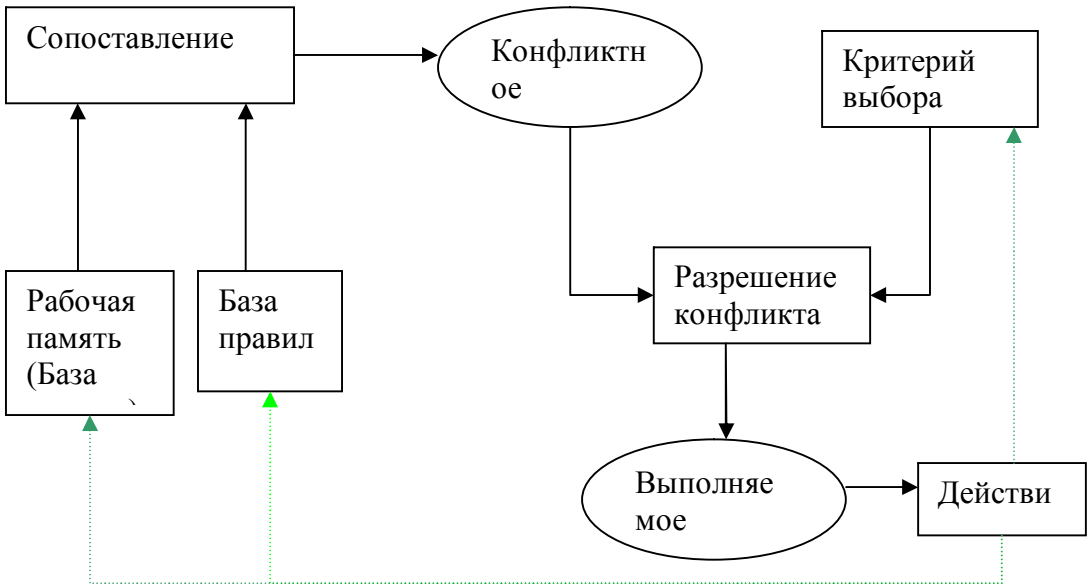


Рис. 3.3. Цикл работы интерпретатора

Информация из рабочей памяти последовательно сопоставляется с посылками правил для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое *конфликтное множество.* Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он выбирает единственное правило, после чего оно срабатывает. Это выражается в занесении фактов, образующих заключение правила, в рабочую память или в изменении критерия выбора конфликтующих правил. Если же в заключение правила входит название какого-нибудь действия, то оно выполняется. Работа машины вывода зависит только от состояния рабочей памяти и от состава базы знаний. На практике обычно учитывается история работать), то есть поведение механизма выпада в предшествующих циклах. Информация о поведении механизма вывода запоминается в памяти состояний (рис. 3.4). Обычно память состояний содержит протокол системы.

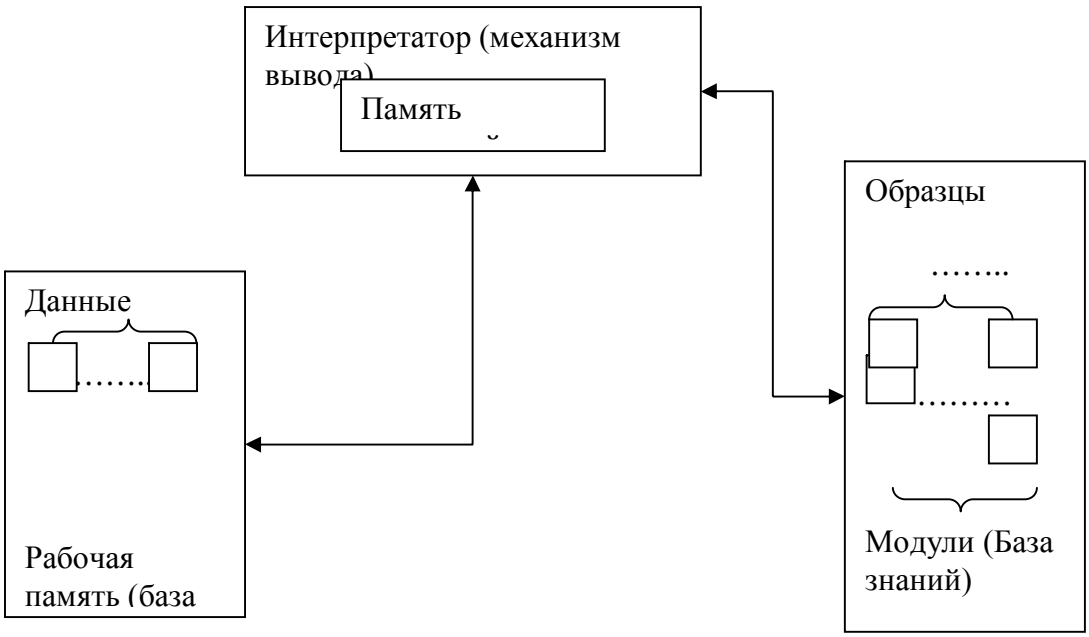


Рис. 3.4. Схема функционирования интерпретатора

От выбранного метода поиска, то есть стратегии вывода, будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно «зашиты» в механизм вывода, поэтому в большинстве систем инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию. При разработке стратегии управления выводом важно определить дна вопроса:

1. Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска — в прямом или обратном направлении,
2. Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора – глубину, в ширину, по подзадачам или иначе.

При обратном порядке вывода вначале выдвигается некоторая гипотеза, а затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу (рис. 3.5. правая часть). Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчинен кой гипотезы. Вывод такого типа называется управляемым целями, или управляемым консеквентами. Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

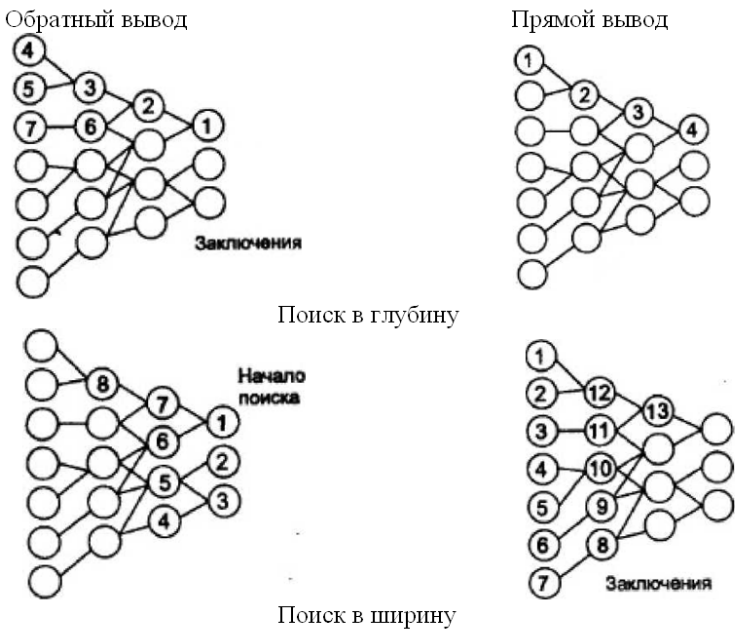


Рис.3.5. Стратегии вывода

В системах с прямым выводом по известным фактам отыскивается заключение, которое из этих фактов следует (см. рис.3.5. левая часть). Если такие заключения удается найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами. Существуют системы, в которых вывод основывается на сочетании упомянутых выше методов — обратного и ограниченного прямого. Такой комбинированный метод получил название циклического.

Пример 3.5

Имеется фрагмент базы знаний на двух правил:

П1. ЕСЛИ «отдых — летом» и «человек — активный», то «ехать в горы».

П2. Если «любит солнца», то «отдых летом».

Предположим, в систему поступили факты – «человек активный» и «любит солнце»,

ПРЯМОЙ ВЫВОД - исходя из фактических данных, получить рекомендацию.

1-й проход.

Шаг 1. Пробуем *П1,* не работает (не хватает данных «отдых— летом»)-

Шаг 2. Пробуем *П2* работает, в базу поступает факт «отдых — летом».

2-й проход:

Шаг 2. Пробуем *П2,* работает, активируется цель «ехать в горы», которая и выступает как совет, который предлагает ЭС.

ОБРАТНЫЙ ВЫВОД - подтвердить выбранную цель при помощи имеющихся правил и данных. 1-й проход.

Шаг 1. Цель — «ехать в горы»: пробуем *П1* — данных «отдых — летом»- нет, они становятся новой целью и пишется правило, где цель в левой части.

Шаг 2. Цель «отдых — летом»: правило *П2* подтверждает цель и активирует ее.

2-й проход:

Шаг 3. Пробуем П1, подтверждается искомая цель.

В системах, база знаний которых насчитывает сотни правил, желательным является использование стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вывода, К числу таких стратегий относятся: поиск в глубину, поиск в ширину, разбиение на подзадачи и альфа-бета алгоритм [Таунсенд, Фохт, 1991; Уэно, Исидзука, 1999; Справочник по ИИ, 1990].

При поиске в глубину в качестве очередной подцели выбирается та, которая соответствует следующему, более детальному уровню описания задачи. Например, диагностирующая система, сделав на основе известных симптомов предположение о наличии определенного заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не опровергнет выдвинутую гипотезу.

При поиске в ширину, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдет к симптомам следующего уровня детальности.

Разбиение на подзадачи — подразумевает выделение подзадач, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Примером, подтверждающим эффективности разбиения на подзадачи, является поиск неисправностей в компьютере — сначала выявляется отказавшая подсистема (питание, память и т. д.), что значительно сужает пространство поиска. Если удается правильно понять сущность задачи и оптимально разбить ее на систему иерархически связанных целей-подцелей, то можно добиться того, что путь к ее решению в пространстве поиска будет минимален.

Альфа-бета алгоритм позволяет уменьшить пространство состояний путем удаления ветвей, неперспективных для успешного поиска. Поэтому просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются. Альфа-бета алгоритм нашел широкое применение в основном в системах, ориентированных на различные игры, например в шахматных программах.

3.3. Нечеткие знания

3.3.1. Основы теории нечетких множеств

При попытке формализовать человеческие знания исследователи вскоре столкнулись с проблемой, затруднявшей использование традиционного математического аппарата для их описаний. Существует целый класс описаний, оперирующих качественными характеристиками объектов (много, мало, сильный, очень сильный и т. п.). Эти характеристики обычно размыты и не могут быть однозначно интерпретированы, однако,

содержат важную информацию (например, «Одним из возможных признаков гриппа является высокая температура»).

Кроме того, в задачах, решаемых интеллектуальными системами, часто приходится пользоваться неточными знаниями, которые не могут быть интерпретированы как полностью истинные или ложные (логические true/false или 0/1). Существуют знания, достоверность которых выражается некоторой промежуточной цифрой, например 0.7.

Как, не разрушая свойства размытости и неточности, представлять подобные знания формально? Для разрешения таких проблем в начале 70-х американский математик *Лотфи Заде* предложил формальный аппарат нечеткой (fuzzy) алгебры и нечеткой логики [Заде, 1972]. Позднее это направление получило широкое распространение [Орловский, 1&S1; Аверкин и др., 1986; Яшин, 1990] и положило начало одной из ветвей ИИ под названием — *мягкие вычисления* (soft computing).

Л. Заде ввел одно ид главных понятий в нечеткой логике — понятие лингвистической переменной.

Лннгвистическая переменная (ЛП) — это переменная, значение которой определяется набором вербальных (то есть словесных) характеристик некоторого свойства.

Например, ЛП *«рост»* определяется через набор *{карликовый, низкий, средний, высокий, очень высокий}.*

Значения лингвистической переменной (ЛП) определяются через так называемые нечеткие множества (НМ), которые в свою очередь определены на некотором базовом наборе значений или базовой числовой шкале, имеющей размерность. Каждое значение ЛП определяется как нечеткое множество (например, НМ « низкий рост»).

Нечеткое множество определяется через некоторую базовую шкалу В и функцию принадлежности НМ — $\mu(x)$, $x \in B$, принимающую значения на интервале [0…1]. Таким образом, нечеткое множество В — это совокупность пар вида $(x, \mu(x))$, где $x \in B$. Часто встречается и такая запись:

$$\sum_{i=1}^{n} \frac{x_i}{\mu(x_i)}$$

где x_i — i-ое значение базовой шкалы.

Функция принадлежности определяет субъективную *степень уверенности* эксперта в том, что данное конкретное значение базовой шкалы соответствует определяемому НМ. Эту функцию не стоит путать с вероятностью, носящей объективный характер и подчиняющейся другим математическим зависимостям.

Например, для двух экспертов определение НМ «высокая» для ЛП «цена автомобиля» в условных единицах может существенно отличаться в зависимости от их социального и финансового положения,

«Высокая_цена_автомобиля_1» ~ {50000/1 + 25000/0.8 + 10000/0.6 + 5000/0.4}

«Высокая_цена_аитомобиля_2» - {25000/1 + 10000/0.8 + 5000/0.7 + 3000/0 4}

Пример 3.6

Пусть перед нами стоит задача интерпретации значений ЛП «возраст», таких как «молодой» возраст, «преклонный» возраст или «переходный» возраст. Определим «возраст» как ЛП (рис. 3.6). Тогда «молодой», «преклонный», «переходный» будут значениями этой лингвистической переменной. Более того, базовый набор значений ЛП следующий: В {младенческий, детский, юный, молодой, зрелый, преклонный, старческий}.

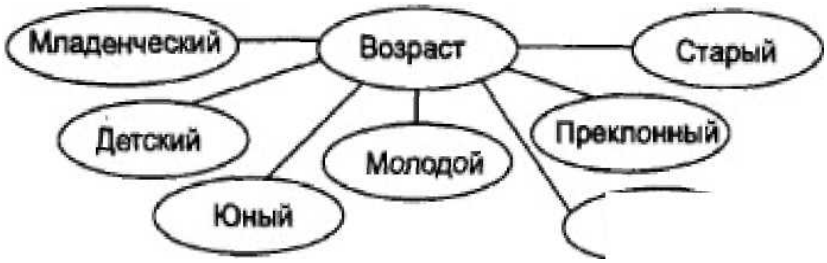


Рис. 3.6. Лингвистическая переменная «возраст» - и нечеткие множества, определяющие ее

Для ЛП «Возраст» базовая шкала — это числовая шкала от 0 до 120, обозначающая количество прожитых лет, а функция принадлежности определяет, насколько мы уверены в том, что данное количество лет можно отнести к данной категории возраста. На рис. 3.7 отражено, как одни и те же значения базовой шкалы могут участвовать в определении различных НМ.

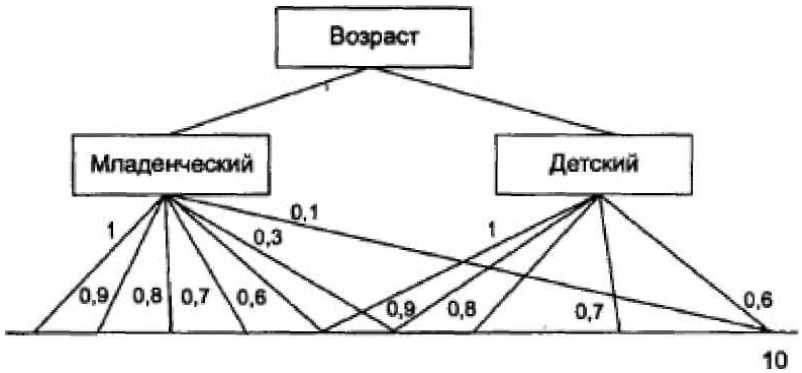


Рис. 3.7. Формирование нечетких множеств

Например, определить значение НМ «младенческий возраст» можно так:

$$\text{«младенческий возраст»} = \left\{ \frac{0.5}{1} + \frac{1}{0,9} + \frac{2}{0,8} + \frac{3}{0,7} + \frac{4}{0,6} + \frac{5}{0,3} + \frac{11}{0,1} \right\}$$

Рис. 3.8 иллюстрирует оценку НМ неким усредненным экспертов, который ребенка до полу года с высокой степенью уверенности относит к младенцам (т - 1). Дети до четырех лет причисляются к младенцам тоже, но с меньшей степенью уверенности ($0.5 < m < 0.9$), а в десять лет ребенка называют так только в очень редких случаях — к примеру, для девяностолетней бабушки и 15 лет может считаться младенчеством. Таким образом, нечеткие множества позволяют при определении понятия учитывать субъективные мнения отдельных индивидуумов.

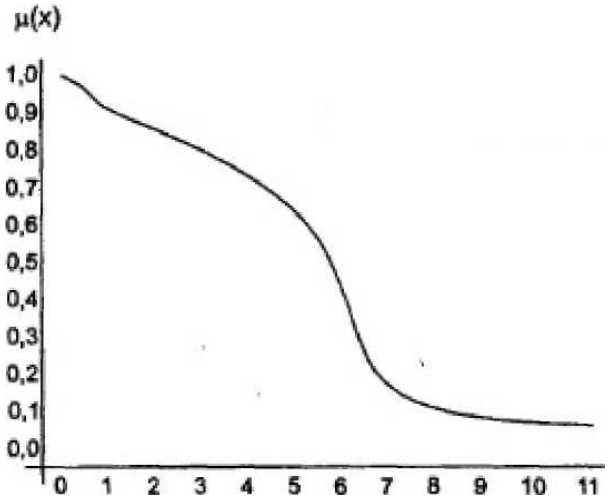


Рис.3.8. График функции принадлежности к нечеткому множеству – младенческий возраст

3.3.2. Операции с нечеткими знаниями

Для операций с нечеткими знаниями, выраженными при помощи лингвистических переменных, существует много различных способов. Эти способы являются в основном эвристиками. Мы не будем останавливаться на этом вопросе подробно, укажем лишь для примера определение нескольких операций, К примеру, операция «ИЛИ» часто задается так [Аверкин и др., 198G; Яшин, 1990]:

(так называемая логика Заде) или так:

(вероятностный подход).

Усиление или ослабление лингвистических понятий достигается введением специальных квантификаторов. Например, если понятие «старческий возраст» определяется как

$$\left\{ \frac{60}{0,6} + \frac{70}{0,8} + \frac{80}{0,9} + \frac{90}{1} \right\}$$

то понятие очень старческий возрасти определится как

$$con\,(A) = A^2 = \sum_i \frac{x_i}{\mu^2},$$

то есть очень старческий возраст равен

$$\left\{ \frac{60}{0,36} + \frac{70}{0,64} + \frac{80}{0,81} + \frac{90}{1} \right\}$$

Для вывода на нечетких множествах используются специальные отношения и операции над ними (подробнее см. работу [Орловский, 1981]). Одним из первых применений теории НМ стало использование коэффициентов уверенности для вывода рекомендаций медицинской системы MYCIN [Shortliffe, 1976]. Этот метод использует несколько эвристических приемов. Он стал примером обработки нечетких знаний, повлиявших на последующие системы, В настоящее время в большинство инструментальных средств разработки систем, основанных на знаниях, включены элементы работы с НМ, кроме того, разработаны

специальные программные средства реализации так называемого нечеткого вывода, например «оболочка» - Fuzzy CLIPS,

3.4. Прикладные интеллектуальные системы

Центральная парадигма интеллектуальных технологий сегодня — это обработка, знаний. Системы, ядром которых является база знаний или модель предметной области, описанная на языке сверхвысокого уровня, приближенном к естественному, называют интеллектуальными. Будем называть такой язык сверхвысокого уровня — *языком представления знаний (ЯПЗ).* Чаще всего интеллектуальные системы (ИС применяются для решения сложных задач, где основная сложность решения связана с использованием слабо-формализованных знаний специалистов- практиков и где логическая (или смысловая) обработка информации превалирует над вычислительной. Например, понимание естественного языка, поддержка принятия решения в сложных ситуациях, постановка диагноза и рекомендации по методам лечения, анализ визуальной информации, управление диспетчерскими пультами и др.

Фактически сейчас прикладные интеллектуальные системы используются в десятках тысяч приложений, а годовой доход от продаж программных и аппаратных средств искусственного интеллекта еще в 1989 г. в США составлял S70 млн. долларов, а и 1990 г. - 14 млрд. долларов [Попов, 1996]. В дальнейшем почти тридцатипроцентный прирост дохода сменился более плавным наращиванием темпов (по материалам [Поспелов, 1997; Хорошевский, 1997; Попов, 1996; Walker, Miller, 1987; Tuthin, 1994, Durkm, 199S]).

На рис. 3.9 отражены различные аспекты состояния рынка искусственного интеллекта: инвестиции в разработку в области ИИ (СШЛ Европа, Япония) (рис. 3.9, а); доля систем ИИ в информатике (программном обеспечении) (рис. 3.9, б)*;* доходы от продаж традиционных языков программирования (рис. 3.9, в); инвестиции только в программное обеспечение (США) (рис. 3.9, г); инвестиции и аппаратное обеспечение (США) (рис. 3.9, д), структура рынка ЭС (США, 1993) (рис, 3.9, е).

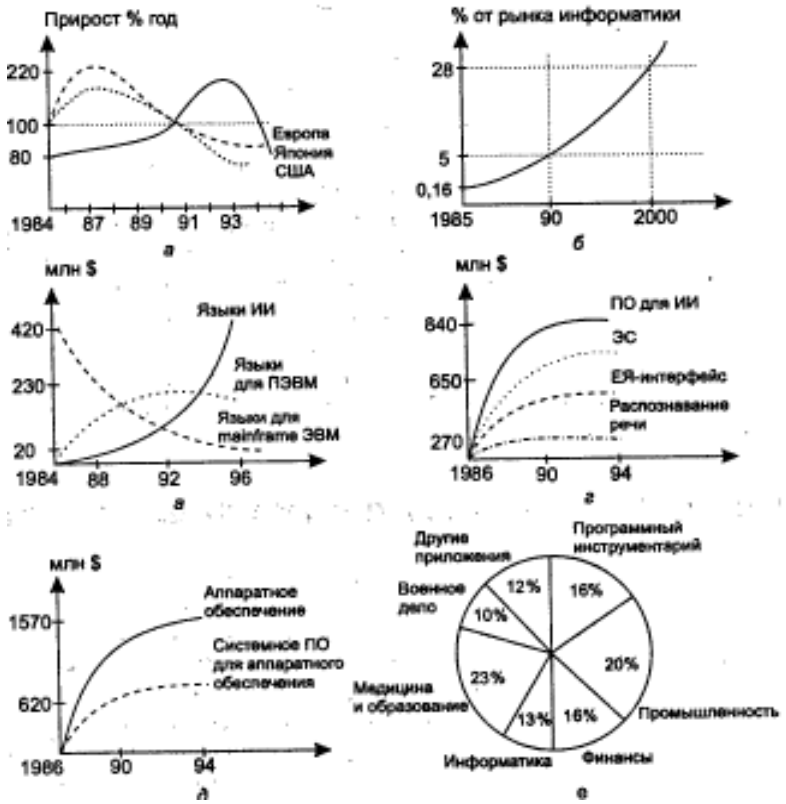


Рис.3.9. Состояние и перспективы рынка ИИ

Наиболее распространенным видом ИС являются экспертные системы.

Экспертные системы (ЭС) — это наиболее распространенный класс ИС, ориентированный на тиражирование опыта высококвалифицированных специалистов в областях, где качество принятии решений традиционно зависит от уровня экспертизы. Например, медицина, юриспруденция, геология, экономике, военное дело и др. ЭС эффективны лишь в специфических «экспертных» областях, где важен эмпирический опыт специалистов.

Только в США ежегодный доход от продаж инструментальных средств разработки ЭС составлял в начале 90-х годоб 300-*А*00 млн. долларов, а от применения ЭС — 80-90 млн. долларов [Поцоч, 1996]. Ежегодно крупные фирмы разрабатывают десятки ЭС типа «in-housc» для внутреннего пользования. Эти системы интегрируют опыт специалистов компании по ключевым и стратегически важным технологиям. В начале 90-х появилась новая наука — *«менеджмент, знаний»* (knowledge management), ориентированная на методы обработки и управления корпоративными знаниями (см. главу 5),

Современные ЭС — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и распространяющие этот эмпирический опыт для консультирования менее квалифицированных пользователей. Разработка экспертных систем, как активно развивающаяся ветвь информатики, направлена на использование ЭВМ дли обработки информации в тех областях науки и техники, где традиционные математические методы моделирования малопригодны. В этих областях важна смысловая и логическая обработка информации, важен опыт экспертов.

Приведем некоторые условия, которые могут свидетельствовать о необходимости разработки и внедрения экспертных систем (частично из [Уотермен, 1989]);

— нехватка специалистов, затрачивающих значительное время для оказания помощи другим;

— выполнение небольшой задачи требует многочисленного коллектива специалистов, поскольку ни один из них не обладает достаточным знанием;

— сниженная производительность, поскольку задача требует полного анализа сложного набора условий, а обычный специалист не в состоянии просмотреть (за отведенное время) все эти условия;

— большое расхождение между решениями самых хороших и самых плохих исполнителей;

— наличие конкурентов, имеющих преимущество в силу того, что они лучше справляются с поставленной задачей.

Подходящие задачи имеют следующие характеристики:

— являются узкоспециализированными;

— не зависят в значительной степени от общечеловеческих знаний или соображений здравого смысла;

— не являются для эксперта ни слишком легким и, ни слишком сложными,(время, необходимое эксперту для решения проблем*ы,* может составлять от трех часов и до трех минут).

Экспертные системы достаточно молоды – первые системы такого рода, MYC1N [Shortliffe, 1976) и DENDRAL [Buchanan, Feigenbaum, 1978], появились в США в середине 70-х годов. В настоящее время в мире насчитывается несколько тысяч промышленных ЭС, которые дают советы:

— при управлении сложными диспетчерскими пультами, например сети распределения электроэнергии, — Alarm Analyser [Walker, Miller, 1987];

— при постановке медицинских диагнозов — ARAMIS [Shortliffe, Buchanan, Feigenbaum, 1979], NEUREX [Reggia, 1988];

— при поиске неисправностей в электронных приборах, диагностика отказов контрольно-измерительного оборудования — Intelligence Ware [Slagle, Gardiner, Kyungsook, 1990], Plant Diagnostics [Уотермен, 1989], FOREST [Finin, McAdams, Kleinosky, 1984];

— по проектированию интегральных микросхем — DAA [Сойер, Фостер, 1988],

NASL [Walker, Miller, 1988], QO [Pega, Sticklen, Bond, 1993];по управлению перевозками — AIR-PLAN [Masui, McDermott, 19831;

— по прогнозу военных действий - ANALYST [Bonasso, 1984], BATTLE [Slagle, Gaynor, 1983];

— по формированию портфеля инвестиций и оценке финансовых рисков — RAD [Kestelyn,1992], налогообложению - RUNE [Durkin, 1998] и т. д.

Наиболее популярные приложения ИС отображены на рис. 3.10 [Durkin, 1998].

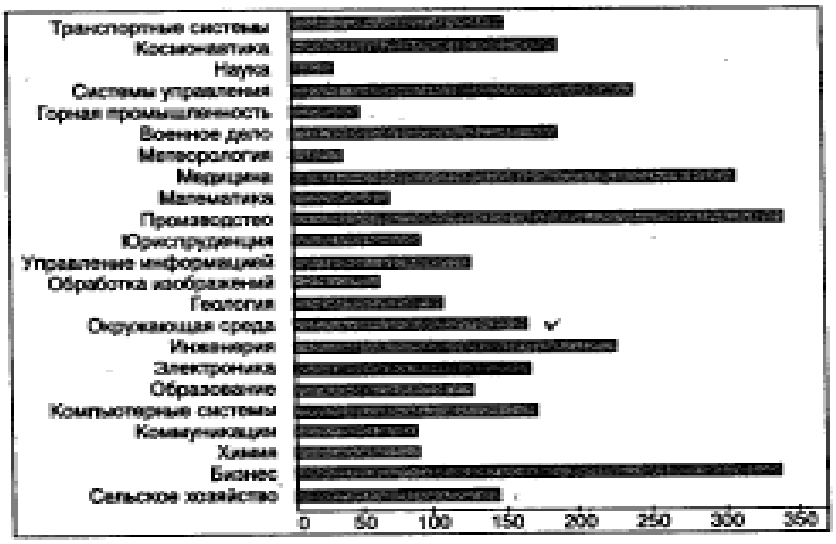


Рис. 3.10. Основные приложения ИС

Сейчас легче назвать области, где еще нет ЭС, чем те, где они уже применяются. Уже в 19Й7 г. опрос пользователей, проведенный журналом "Intelligent Technologies" (США), показал, что примерно:

— 25 % пользователей используют ЭС;

— 25 % собираются приобрести ЭС в ближайшие 2-3 года;

— 50 % предпочитают npoeectn исследование об эффективности их использования.

Главное отличие ИС и ЭС от других программных средств - это наличие базы знаний (БЗ), в которой знания хранятся в форме, понятной специалистам предметной области, и могут быть изменены и дополнены также в понятной форме. Это и есть языки представления знаний — ЯПЗ.

До последнего времени именно различные ЯПЗ были центральной проблемой при разработке ЭС. Сейчас существуют десятки языков или моделей представления знаний (см. параграф 1.2.2). Наибольшее распространение получали следующие модели:

— продукции (OPS5 [Forgy, 1981]; ROSIE [Fain, Hayes-Roth, Sowizral, Waterman, 1982])

— семантические сети (SIMER+MIR [Осипов, 1997]; NET [Цейтин, 1985] ;

— фреймы (FRL [Байдун, Бунин, 1988; Справочник по ИИ, 1990]);;

— логическое программирование (ПРОЛОГ [Макалистер, 1990; Стерлинг, Шапиро, 1990]);

— объектно-ориентированные языки (SMALLTALK [Goldberg, Robson, 1983; Буч, 1993]; CLOS [Pega, Sticklen, Bond, 1993]).

Для перечисленных выше моделей существует соответствующая математическая нотация, разработаны системы программирования, реализующие эти ЯПЗ. И имеется большое количество реальных коммерческих ЭС. Подробнее вол росы программной реализации прикладных ИС рассмотрены в главе 6. Современное состояние разработок в области ЭС в России можно охарактеризовать как стадию все возрастающего интереса среди широких слоев специалистов - финансистов, топ-менеджеров, преподавателей, инженеров, медиков,

психологов, программистов$_1$ лингвистов. В последние годы этот интерес имеет пока достаточно слабое материальное подкрепление - явная нехватка учебников и специальной литературы, отсутствие символьных процессоров и рабочих станций, ограниченное финансирование исследований в этой области, слабый отечественный рынок программных продуктов для разработки ЭС. Поэтому появляется возможность распространения «подделок» под экспертные системы в виде многочисленных диалоговых систем и интерактивных пакетов прикладных программ, которые дискредитируют в глазах пользователей это чрезвычайно перспективное направление. Процесс создания экспертной системы требует участия высококвалифицированных специалистов в области искусственного интеллекта, которых пока выпускает небольшое количество высших учебных
заведений страны.

Наибольшие трудности в разработке ЭС вызывает сегодня не процесс машинной реализации систем, а домашинный этап анализа знаний и проектирования базы знаний. Этим занимается специальная наука - инженерия знаний [Гаврилова, Червинская, 1992; Adeli, 1994; Scott, Clayton, Gibson, 1994]. Современному состоянию этой науки и посвящены последующие главы этой книги.

3.5. Разработка систем, основанных на знаниях

3.5.1. Введение в экспертные системы. Определение и структура

В качестве рабочего определения экспертной системы примем следующее.

Экспертные системы (ЭС) — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных области и тиражирующие этот эмпирический опыт для консультаций менее квалифицированные пользователей.

Обобщенная структура экспертной системы представлена на рис. 3.11. Следует учесть, что реальные ЭС могут иметь более сложную структуру, однако блоки, изображенные на рисунке, непременно присутствуют в любой действительно экспертной системе, поскольку представляют собой стандарт de facto структуры современной ЭС.

В целом процесс функционирования ЭС можно представить следующим образом: пользователь, желающий получить необходимую информацию через пользовательский интерфейс, посылает запрос к ЭС; решатель, пользуясь базой знаний, генерирует и выдает пользователю подходящую рекомендацию, объясняя ход своих рассуждений при помощи подсистемы объяснений.

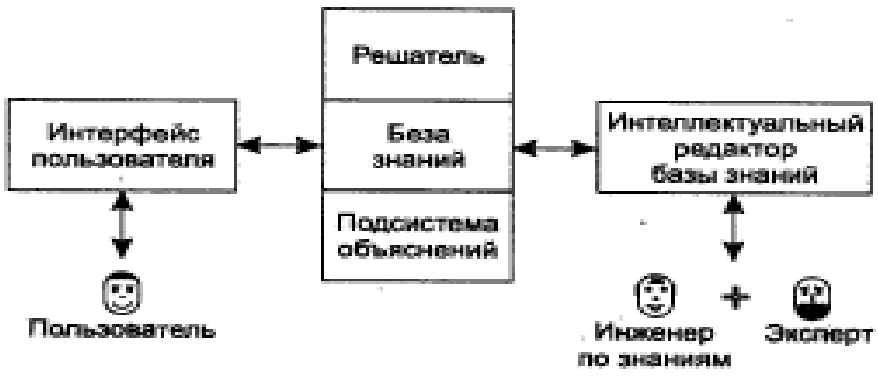


Рис. 3.11. Структура экспертной системы

Так как терминология в области разработки ЭС постоянно модифицируется, определим основные термины в рамках данной работы.

Пользователь — специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям — специалист в области искусственного интеллекта, выступающий в роли промежуточного буфера менаду экспертом и базой знаний. Синонимы: *криптолог, инженер-интерпретатор, аналитик.*

Интерфейс пользователя — комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.

База знаний (53) — ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.

Решатель — программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: *дедуктивная машина, машина вывода, блок логического вывода,*

Система объяснений — программа, позволяющая пользователю получить ответ на вопросы: «Как была получена та или иная рекомендация?» и «Почему и прияла такое решение?». Ответ на вопрос «Как?» — это трактовка всего получения решения с указанием использованных фрагментов БЗ, то есть всех шагов цепи умозаключений. Ответ на вопрос «Почему?» — ссылка на умозаключение непосредственно предшествовавшее полученному решению, то есть отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.

Интеллектуальный редактор БЗ — программа, представляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок («help» — режим) и других сервисных средств, облегчающих работу с базой.

Еще раз следует подчеркнуть, что представленная на рис. 3.11 структура является минимальной, что означает обязательное присутствие указанных на ней блоков. Если система объявлена разработчиками как экспертная, только наличие всех этих блоков гарантирует реальное использование аппарата обработки знаний. Однако промышленные прикладные ЭС могут быть существенно сложнее и дополнительно включать базы данных, интерфейсы обмена данными с различными пакетами прикладных программ, электронными библиотеками и.т.д.

3.5.2. Классификация систем, основанных на знаниях

Класс ЭС сегодня объединяет несколько тысяч различных программных комплексов, которые можно классифицировать по различным критериям. Полезными могут оказаться классификации, представленные на рис. 3.12.

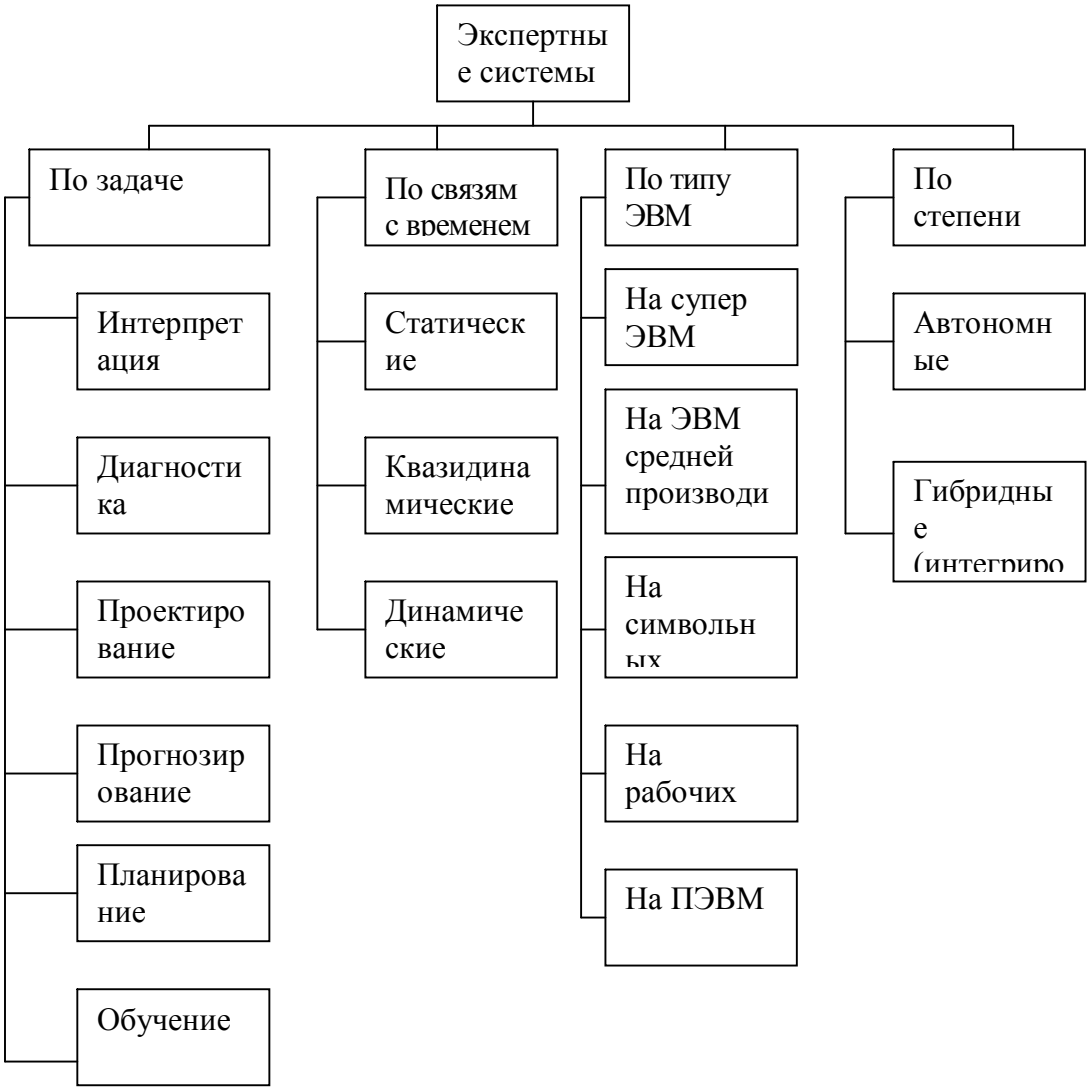


Рис. 3.12. Классификация экспертных систем

Рассмотрим указанные на рисунке типы задач подробнее.

— Интерпретация данных. Это одна из традиционных задач для экспертных систем. Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.

Все примеры далее из работ (Попов и др. (ред.), 1990; Соловьев, Соловьева, 1989; Хейес-Рот и др., 1987].

— Обнаружение и идентификация различных типов океанских судов по результатам аэрокосмического сканирования — SIAP;

— Определение основных свойств личности по результатам психодиагностического тестирования в системах АВТАНТЕСТ и МИКРОЛЮШЕР и др.

— Диагностика. Под диагностикой понимается процесс соотнесения объекта с некоторым классом объектов и/или обнаружение неисправности в некоторой системе. Неисправность - это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Важной спецификой является здесь необходимость понимания функциональной структуры («анатомии») диагностирующей системы.

— Диагностика и терапия сужения коронарных сосудов — ANGY;

— Диагностика ошибок в аппаратуре и математическом обеспечении *ЭВМ,*

система CRIB и др.

— • Мониторинг. Основная задача мониторинга — непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или и них параметров за допустимые пределы. Главные проблемы — «пропуск» тревожной ситуации и инверсная задача «ложного» срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимость учета временного контекста.

— • Контроль за работой электростанций и помощь диспетчерам атомного реактора – REACTOR;

— Контроль аварийных датчиков на химическом заводе — FALCON и др.

— • Проектирование. Проектирование состоит в подготовке спецификаций на создание «объектов» с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов - чертеж, пояснительная записка и т.д. Основные проблемы здесь — получение четкого структурного описания знаний об объекте и проблема «следа». Для организации эффективного проектирования и в еще большей степени перепроектирования необходимо формировать не только сами проекты решения, но и мотивы их принятия. Таким образом в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ЭС; процесс вывода решения и процесс объяснения.

— Проектирование конфигураций ЭВМ VAX — 11/780 в системе XCON (или R1), проектирование БИС — CADHELP;

— Синтез электрических цепей — SYN.

— Прогнозирование. Прогнозирование позволяет предсказывать последствия некоторых событий или явлений как основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию, Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками.

— Предсказание погоды — система WILLARD.

— Оценки будущего урожая — PLANT.

— Прогнозы в экономике — ECON и др.

— Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

— Планирование поведения робота — STRIPS.

— Планирование промышленных заказов — ISIS.

— Планирование эксперимента — MOLGFN и др.

— Обучение. Под обучением понимается использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом «ученике» и его характерных ошибках, затем в работе они способны диагностировать слабости в познаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

— Обучение языку программирования ЛИСП в системе «Учитель Лиспа»;

— Система PROUST - обучение языку Паскаль и др.

— • Управление. Под управлением понимается функция организованной системы, поддерживающая определенный резким деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями.

— Помощь в управлении газовой котельной — GAS.

— Управленце системой календарного планирования Project Assistant и др.

— • Поддержка принятия решении. Поддержка принятия решения - это совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающими процесс принятия решения. Эти ЭС помогают специалистам выбрать и/или сформировать нужную альтернативу среди множества выборов при принятии ответственных решений.

— Выбор стратегии выхода фирмы из кризисной ситуации — CRYSIS;

— Помощь в выборе страховой компании или инвестора - CHOICE и др.

В общем случае все системы, основанные на знаниях, можно подразделить на *системы, решающие задачи анализа,* и на *системы, решающие задачи синтеза.* Основное отличие задач анализа от задач синтеза заключается в том, что если в задачах анализа множество решений может быть перечислено и включено в систему, то в задачах синтеза множество решений потенциально не ограничено и строится из решений компонент или подпроблем. Задачами анализа являются: интерпретация данных, диагностик поддержка принятия решения; к задачам синтеза относятся проектирование, планирование, управление. Комбинированные: обучение, мониторинг, прогнозирование.

По связи с реальным временем выделяют:

— *Статические* ЭС разрабатываются и предметных областям в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны. Пример: диагностика неисправностей в автомобиле.

— *Квазидинамические* ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Пример: микробиологические ЭС, и которых снимаются лабораторные измерения технологического процесса один раз в 4-5 часов и анализируется диаграмм и полученных показателей по отношению к предыдущему измерению.

— *Динамические ЭС* работают а сопряжении с датчиками объектив в режиме реального времени с непрерывной интерпретацией поступающих в систему данных. Примеры: управление гибкими производственными комплексами, мониторинг в реанимационных палатах; программный инструментарий для разработки динамических систем- G2 [Попов,].

На сегодняшний день в зависимости от типа ЭВМ существуют:

— ЭС для уникальных стратегически важных задач на суперЭВМ (Эльбрус, CRAY, CONVEX и др.);

— ЭС на ЭВМ средней производительности (типа ЕС ЭВМ, mainframe);

— ЭС на символьных процессорах и рабочих станций (SUN, APOLLO);

— ЭС на мини- и супермини-ЭВМ (VAX, micro-VAX и др.);

— ЭС на персональных компьютерах (IBM PC, MAC II и подобные).

По степени интеграции с другими программами выделяют следующие экспертные системы:

Автономные ЭС работают непосредственно в режиме консультаций с пользователем для специфически «экспертных» задач, для решения которых не требуется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.).

Гибридные ЭС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Это может бить интеллектуальная надстройка над ППП (пакетами прикладных программ) или интегрированная среда для решения сложной задачи с элементами экспертных знаний.

Несмотря на внешнюю привлекательность гибридного подхода, следует отметить, что разработка таких систем являет собой задачу на порядок более сложную, чем разработка автономной ЭС. Стыковка не просто разных пакетов, а разных методологий (что происходит в гибридных системах) порождает целый комплекс теоретических и практических трудностей.

3.6. Коллектив разработчиков

Под коллективом разработчиков (КР) будем понимать группу специалистов, ответственных за создание ЭС.

Как видно из рис. 3.11, в состав КР входят по крайней мере три человека - пользователь, эксперт и инженер по знаниям. На рисунке не видно программиста. Таким образом, минимальный состав КР включает четыре человека; реально же он разрастается до 8-10 человек. Численное увеличение коллектива разработчиков происходит по следующим причинам: необходимость учета мнения нескольких пользователей, помощи нескольких экспертов; потребность как в проблемных, так и системных программистах. На Западе в этот коллектив дополнительно традиционно включают менеджера и одного технического помощника.

Если использовать аналогии из близких областей, то КР более всего схож с группой администратора базы данных при построении интегрированных информационных систем или бригадой программистов, разрабатывающих сложный программный комплекс. При отсутствии профессионального менеджера руководителем КР, участвующим во всех стадиях разработки, является инженер по знаниям, поэтому к его квалификации предъявляются самые высокие требования.

Для обеспечения эффективности сотрудничества любой творческой группы, в том числе и группы КР ЭС, необходимо возникновение атмосферы взаимопонимания и доверия, которое, в свою очередь, обусловлено психологической совместимостью членов группы; следовательно, при формировании КР лоджии учитываться психологические свойства участников.

В настоящий момент в психологии существует несколько десятков методик по определению свойств личности, широко используемых и вопросах профессиональной ориентации. Эти психодиагностические методики, часть на которых уже автоматизирована, различаются направленностью, глубиной, временем опроса и способами интерпретации. В частности, система АВАНТЕСТ (Автоматический Анализ Тестов) [Гаврилова, 1984] позволяет моделировать рассуждения психолога при анализе результатов тестирования по 16-факторному опроснику Р. Кэттелла и выдает связное психологическое заключение на естественном русском языке, характеризующее такие свойства личности, как общительность, аналитичность, добросовестность, самоконтроль и т. п. Модифицированная база знании системы АВТАНТЕСТ позднее была использована в ЭС «Cattell» (см. параграф 7.3).

Ниже приведены два аспекта характеристик членов КР: 1 — психофизиологический 2 — профессиональный.

Пользователь

1. К пользователю предъявляются самые слабые требования, поскольку его не выбирают. Он является в некотором роде заказчиком системы. Желательные качества:

а) дружелюбие;
б) умение объясните что же он хочет от системы;
в) отсутствие психологического барьера к применению вычислительной техники;
г) интерес к новому.

От пользователя зависит, будет ли применяться разработанная ЭС, Замечено, что наиболее ярко такие качества проявляются в молодом возрасте, поэтому иногда такие пользователи охотнее применяют ЭС, не испытывая при этом комплекса неполноценности оттого, что ЭВМ им что-то подсказывает.

2. Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС. Кроме того, должна быть полная совместимость в терминологии интерфейса к ЭС с той, которая привычна и удобна для пользователя. Обычно требования к квалификации пользователя не очень велики, иначе он переходит в разряд экспертов и совершенно не нуждается в ЭС:

Эксперт

1. Эксперт — чрезвычайно важная фигура в группе КР. В конечном счете, его подготовка определяет уровень компетенции базы знаний. Желательные качества:

а) доброжелательность;

б) готовность поделиться своим опытом;

в) умение объяснить (педагогические навыки);

г) заинтересованность (моральная, а лучше еще и материальная) в успешности разработки.

Возраст эксперта обычно почтенный, что необходимо учитывать всем членам группы. Часто встает вопрос о количестве экспертов. Поскольку проблема совмещения подчас противоречивых знаний остается открытой, обычно с каждым из экспертов работают индивидуально, иногда создавая альтернативные базы.

2. Помимо безусловно высокого профессионализма в выбранной предметной области, желательно знакомство эксперта с популярной литературой по искусственному интеллекту и экспертным системам для того, чтобы эффективнее прошел этап извлечения знаний.

Программист

1. Известно, что программисты обладают самой низкой потребностью и общении среди представителей разных профессий. Однако при разработке ЭС необходим тесный контакт членов группы, поэтому желательны следующие его качества:

а) общительность;

б) способность отказаться от традиционных навыков и освоить новые методы;

в) интерес к разработке.

2. Поскольку современные ЭС – сложнейшие и дорогостоящие программные комплексы, программисты в КР должны иметь опыт и навыки разработки программ. Обязательно знакомство с основными структурами представления знаний и механизмами вывода, состоянием отечественного и мирового рынка программных продуктов для разработки ЭС и диалоговых интерфейсов.

Инженер по знаниям

Существуют такие профессии и виды деятельности, для которых природные качества личности (направленность, способности, темперамент) могут иметь характер абсолютного показания или противопоказания к занятиям. По-видимому, инженерия знаний принадлежит к таким профессиям. По различным оценкам это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей. Попытаемся дать наброски к портрету инженера по знаниям (без претензии на пехоту и точность определений). *Пол.* Психологи утверждают, что мужчины более склонны к широкому охвату явлений и в среднем у них выше аналитичность, чрезвычайно полезная инженеру по знаниям, которому надо иметь развитое логическое мышление и умение оперировать сложными формальными структурами. Кроме того, при общении с экспертами, которые в большинстве своем настроены скептически по отношению к будущей ЭС. инженер по знаниям - мужчина вызывает более высокую мотивацию успешности со стороны эксперта - женщины. С другой стороны, известно, что у женщин выше наблюдательность к отдельным деталям объектов. Так что пол не является окончательным показанием или противопоказанием к данной профессии.

Это понятие вызывает самые бурные споры специалистов: существует до 50 определений интеллекта, но с прагматической точки зрения, очевидно, что специалист в области искусственного интеллекта должен стремиться к максимальным оценкам по тестам как вербального, так и невербального интеллекта.

Стиль общения. Инженер по знаниям «задает тон» в общении с экспертом, он ведет диалог, и от него в конечном счете зависит его продуктивность. Можно выделить два стиля общения: деловой (или жесткий) и дружеский (или мягкий, деликатный). Нам кажется, что дружеский будет заведомо более успешным, так как снижает «эффект фасада» у эксперта, раскрепощает его. Деликатность, внимательность, интеллигентность, ненавязчивость, скромность, умение слушать и задавать вопросы, хорошая коммуникабельность и в то же время уверенность в себе - вот рекомендуемый стиль общения. Безусловно, что это дар и искусство одновременно, однако занятия по психологическому тренингу могут дать полезные навыки.

Портрет инженера по знаниям можно было бы дополнить другими характеристиками - широтой взглядов и интересов, артистичностью, чувством юмора, обаянием и т. д.

При определении профессиональных требований к аналитику следует учитывать, что ему необходимы различные навыки и умения для грамотного и эффективного проведения процессов извлечения, концептуализации и формализации знаний

Инженер по знаниям имеет дело со всеми формами знаний (см. параграф 1.3).

Z1 (знания в памяти) - Z2 (знания в книгах) - Z3 (поле знаний) - 24 (модель знаний)- Z5 (база знаний).

Работа на уровне ZI требует от инженера по знаниям знакомства с элементами когнитивной психологии и способами репрезентации понятий и процессов в памяти человека, с двумя основными механизмами мышления — ЛОГИЧЕСКИМ И ассоциативным, с такими способами активизации мышления как игры, мозговой штурм и др. с различными моделями рассуждений.

Изучение и анализ текстов на уровне Z2 подразумевает широкую общенаучную подготовку инженера; знакомство с методами реферирования и аннотирования текстов; владение навыками быстрого чтения, а также текстологическими методами извлечения знаний.

Разработка поля знаний на уровне Z3 требует квалифицированного знакомства с методологией представления знаний, системным анализом, теорией познания, аппаратом многомерного шкалирования, кластерным и факторным анализом. Разработка формализованного описания Z4 предусматривает предварительное изучение аппарата математической логики и современных языков представления знаний. Модель знаний разрабатывается на основании результатов глубокого анализа инструментальных средств разработки ЭС и имеющихся «оболочек». Кроме того, инженеру познаниям необходимо владеть методологией разработки ЭС включая методы быстрого прототипирования.

И наконец, реализация базы знаний Z5, в которой инженер по знаниям участвует вместе с программистом, подразумевает овладение практическими навыками работы на ЭВМ и, возможно, одним из языков программирования.

Так как инженеров по знаниям «выращивают» из программистов, уровень Z5 обычно не вызывает затруднения, особенно если разработка ведется на традиционных языках типа С или Паскаль. Специализированные языки искусственного интеллекта ЛИСП и Пролог требуют некоторой перестройки архаично-алгоритмического мышления.

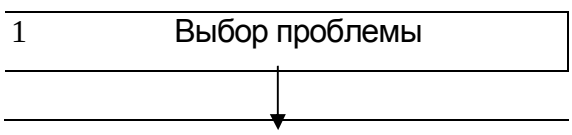
Успешность выбора и подготовки коллектива разработчиков ЭС определяет эффективность и продолжительность всего процесса разработки.

3.7. Технология проектирования и разработки

3.7.1. Проблемы разработки промышленных ЭС

Разработка программных комплексов экспертных систем как за рубежом, так и в нашей стране находится на уровне скорее искусства, чем науки. Это связано с тем, что долгое время системы искусственного интеллекта внедрялись в основном во время фазы проектирования, а чаще всего разрабатывалось несколько прототипных версий программ, и на их основе уже создавался конечный продукт. Такой подход действует хорошо в исследовательских условиях, однако в коммерческих условиях он является слишком дорогим, чтобы оправдать затраты на разработку.

Процесс разработки промышленной экспертной системы, опираясь на традиционные технологии [Николов и др. 1990; Хейес-Рот и др., 1987], практически для любой предметной области можно разделить на шесть более или менее независимых этапов (рис. 3.13).



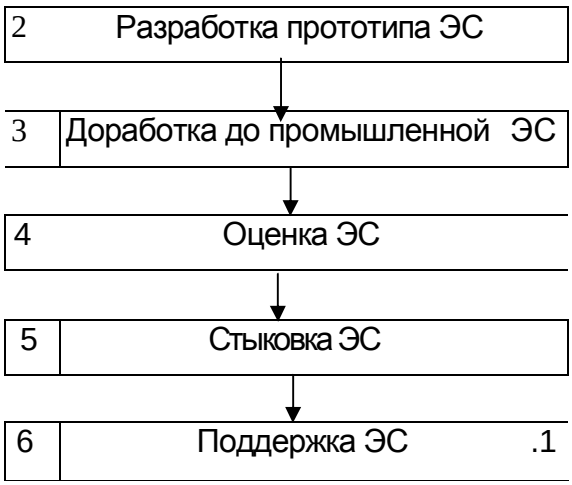


Рис. 3.13 Этапы разработки ЭС

Последовательность этапов дана только с целью получения общего представления о процессе создания идеального проекта. Конечно, последовательность эта не вполне фиксированная. В действительности каждый последующий этап разработки может принести новые идеи, которые могут повлиять на предыдущие решения и даже привести к их переработке. Именно поэтому многие специалисты по информатике весьма критично относятся к методологии экспертных систем. Они считают, что расходы на разработку таких систем очень большие, время разработки слишком велико, а полученные в результате программы накладывают тяжелое бремя на вычислительные ресурсы.

В целом за разработку экспертных систем целесообразно браться организации, где накоплен опыт по автоматизации рутинных процедур обработки информации, таких как:

- формирование корпоративных информационных систем;
- организация сложных расчетов;
- работа с компьютерной графикой;
- обработка текстов и автоматизированный документооборот. Решение таких задач, во-первых, подготавливает высококвалифицированных специалистов по информатике, необходимых для создания интеллектуальных систем, во-вторых, позволяет отделить от экспертных систем неэкспертные задачи.

3.7.2. Выбор подходящей проблемы

Этот этап определяет деятельность, предшествующую решению начать разрабатывать конкретную ЭС. Он включает [Николов и др., 1990]:

- определение проблемной области и задачи;
- нахождение эксперта, желающего сотрудничать при решении проблемы, и на значение коллектива разработчиков;
- определение предварительного подхода к решению проблемы;
- анализ расходов и прибылей от разработки;
- подготовку подробного плана разработки.

Правильный выбор проблемы представляет самую критическую часть разработки в целом. Если выбрать неподходящую проблему, можно очень бистро увязнуть в «болоте» проектирования задач, которые никто не знает, как решать. Неподходящая проблема может также привести к созданию экспертной системы, которая стоит намного больше, чем экономит. Дело будет обстоять еще хуже, если разработать систему, которая работает, но неприемлема для пользователей. Даже если разработка выполняется самой организацией для собственных целей, эта фаза является подходящим моментом для получений: рекомендаций извне, чтобы гарантировать удачно выбранный и осуществимый с технической точки зрения первоначальный проект.

При выборе области применения следует учитывать, что если знание, необходимое для решения задач, постоянное, четко формулируемое и связанное с вычислительной обработкой, то обычные алгоритмические программы, по всей вероятности, будут самым целесообразным способом решения проблем в этой области.

Экспертная система ни в коем случае не устранит потребность в реляционных базах данных, статистическом программном обеспечении, электронных таблицах и системах текстовой обработки. Но если результативность задачи зависит от знания, которое является субъективным, изменяющимся, символьным или вытекающим частично из соображений здравого смысла, тогда область может обоснованно выступать претендентом на экспертную систему.

Обычно экспертные системы разрабатываются путем получения специфических знаний от эксперта и ввода их в систему, Некоторые системы могут содержать стратегии одного индивида. Следовательно, найти подходящего эксперта — это ключевой шаг в созданий экспертных систем.

В процессе разработки и последующего расширения системы инженер по знаниям и эксперт обычно работают вместе. Инженер по знаниям помогает эксперту структурировать знания, определять и формализовать понятия и правила, необходимые для решения проблемы.

Во время первоначальных бесед о ни должны решить, будет ли их сотрудничество успешным. Это немаловажно, поскольку обе стороны будут работать совместно по меньшей мере в течение одного года. Кроме них в коллектив разработчиков целесообразно включить потенциальных пользователей и профессиональных программистов. Подробно функции каждого члена коллектива описаны в следующем параграфе.

Предварительный подход к программной реализации задачи определяется исходя из характеристик задач и ресурсов, выделенных на ее решение. Инженер по знаниям выдвигает обычно несколько вариантов, связанных с использованием имеющихся на рынке программных средств. Окончательный выбор возможен лишь на этапе разработки прототипа.

После того как задача определена, необходимо подсчитать расходы и прибыль от разработки экспертной системы, В расходы включаются затраты на оплату труда коллектива разработчиков, В дополнительные расходы будет включена стоимость приобретаемого программного инструментария, с помощью которого будет разработана экспертная система.

Прибыль может быть получена за счет снижения цены продукции, повышения производительности труда, расширения номенклатуры продукции или услуг или даже разработки новых видов продукции или услуг в области в которой будет использоваться ЭС,

Соответствующие расходы и прибыль от системы определяются относительно времени, в течение которого возвращаются средства, вложенные в разработку. На современном этапе большая часть фирм, развивающих крупные экспертные системы предпочли разрабатывать дорогостоящие проекты, приносящие значительную прибыль.

Можно ожидать развития тенденции разработки менее дорогостоящих систем, хотя и с более длительным сроком окупаемости вложенных в них средств, так как программные средства разработки экспертных систем непрерывно совершенствуются. После того как инженер по знаниям убедился что:

- данная задача может быть решена с помощью экспертной системы;
- экспертную систему можно создать предлагаемыми на рынке средствами;
- имеется подходящий эксперт;
- предложенные критерии производительности являются разумными;
- затраты и срок их окупаемости приемлемы для заказчика

Он составляет план разработки. План определяет шаги процесса разработки и необходимые затраты, а также ожидаемые результаты.

3.7.3. Технология быстрого прототипирования

Прототипная система является усеченной версией экспертной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта. Она также дает возможность инженеру по знаниям привлечь эксперта к активному участию в процессе разработки экспертной системы, и, следовательно, к принятию им обязательства приложить все усилия к оснащению системы в полном объеме.

Объем прототипа — несколько десятков правил, фреймов или примеров. На рис. 3.14 изображено шесть стадий разработки прототипа и минимальный коллектив разработчиков, занятых на каждой из стадий (пять стадий заимствовано из работы [Хейес-Рот и др., 1987]), Приведем кратную характеристику каждой из стадий, хотя эта схема представляет собой грубое приближение к сложному, итеративному процессу.

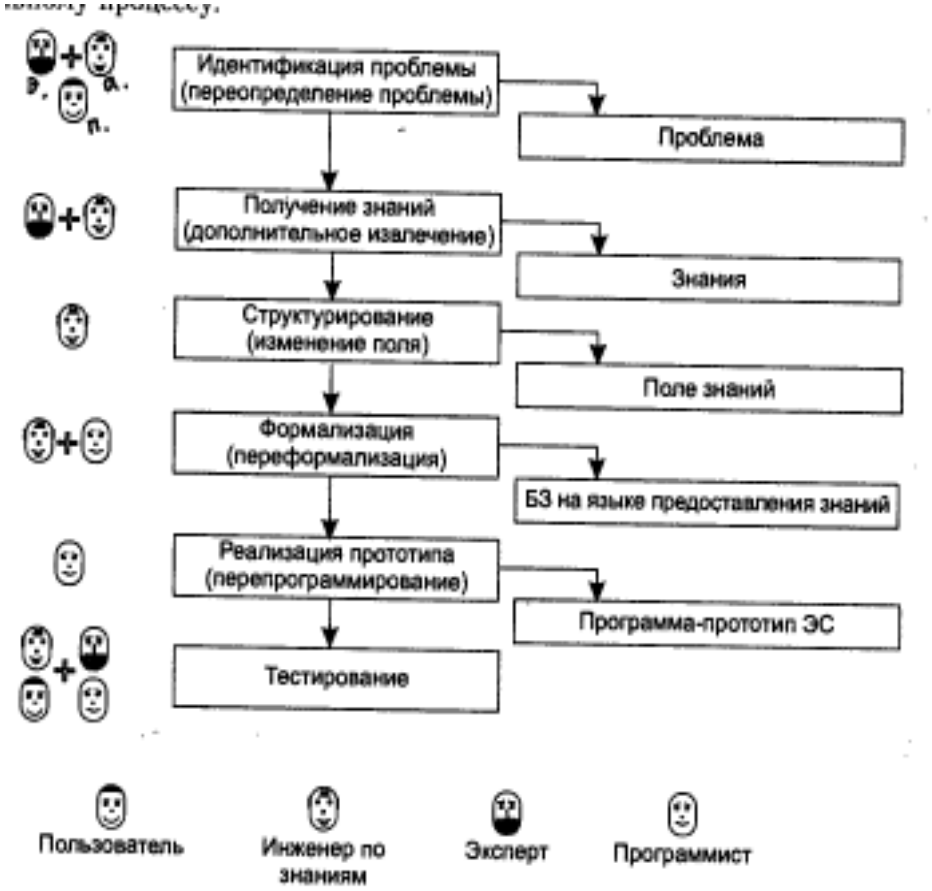


Рис. 3.14. Стадии разработки прототипа ЭС

Хотя любое теоретическое разделение бывает часто условным, осознание коллективом разработчиков четких задач каждой стадии представляется целесообразным. Роли разработчиков (эксперт, программист, пользователь и аналитик) являются постоянными на протяжении всей разработки. Совмещение ролей нежелательно.

Сроки приведены условно, так как зависят от квалификации специалистов и особенностей задачи.

1. Идентификация проблемы. Уточняется задача, планируется ход разработки прототипа экспертной системы, определяются:

- необходимые ресурсы (время, люди$_1$ ЭВМ и т. д.);
- источники знаний (книги, дополнительные эксперты, методики);
- имеющиеся ан алогичные экспертные системы;
- цели (распространение опыта, автоматизация рутинных действий и др.);
- классы решаемых задач и т.д.

Идентификация проблемы — знакомство и обучение членов коллектива разработчиков, в также создание неформальной формулировки проблемы.

Средняя продолжительность 1-2 недели.

2. Извлечение знаний. На этой стадии происходит перенос компетентности от эксперта к инженеру по знаниям, с использованием различных методов (см. главу 4):

- анализ текстов;
- диалоги;
- экспертные игры;
- лекции;
- дискуссии;
- интервью;
- наблюдение и другие.

Извлечение знаний — получение инженером по знаниям наиболее полного из ёиможных представлений о предметной области и способах принятий, решения в ней.

Средняя продолжительность 1-3 месяца.

3. Структурирование или концептуализация знаний. Выявляется структура полученных знании о предметной области, то есть определяются:

- терминология;
- список основных понятий и их атрибутов;
- отношения между понятиями;
- структура входной и выходной информации;
- стратегия принятий решений;
- ограничения стратегий и т. д.

Структурирование или концептуализация знаний - разработка неформального описания знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области.

Такое описание называется *полем знаний.* Средняя продолжительность этапа 2-4 недели. Подробно стадия структурирования описана далее.

4. Формализация. Строится формализованное представление концепций предметной области на основе выбранного языка представления знаний (ЯПЗ). Традиционно на этом этапе используются:

- логические методы (исчисления предикатов 1-го порядка и др.);
- продукционные модели (с прямым и обратным выводом);
- семантические сети;
- фреймы;
- объектно-ориентированные языки, основанные на иерархии классов, объектов.

Разработка баз знаний на языке преставления знаний, который, с одной стороны, соответствует структуре поля знаний, а с другой – позволяет реализовать прототип системы на следующей стадии программной реализации.

Все чаще на этой стадии используется симбиоз языков представления знаний, например, в системе ОМЕГА [Справочник по ИИ, 1990] - фреймы - семантические сети - полный набор возможностей языка исчисления предикатов. Средняя продолжительность 1-2 месяца. Подробно см. в главах 3, 4.

Реализация. Создается прототип экспертной системы, включающий базу знаний и остальные блоки, при помощи одного из следующих способов:

- программирование на традиционных языках типа Pascal, C++ и др.;
- программирование на специализированных языках, применяемых в задачах искусственного интеллекта: LISP [Хювенен, Сеппянен, 1991], FRL [Байдун, Бунин, 1990], SMALLTALK [Справочник по ИИ, 199GJ и др.;
- использование инструментальных средств разработки ЭС типа СПЭИС [Ковригин, Перфильев, 198S], ПИЭС [Хорошевский, 1993], G2 [Попов, Фомины, Кисель, 1996);

• использование «пустых» ЭС или «оболочек» типа ЭКСПЕРТ [Кирсанов, Попов, 1990], ФИАКР [Соловьев, Солоньева, 1989] и др.

Реализация — разработка программного комплекса, демонстрирующего жизнеспособность подхода в целом. Чаще всего первый прототип отбрасывается на этапе реализации действующей.

Средняя продолжительность 1-2 месяца. Более подробно эти вопросы рассматриваются в главе 6.

Тестирование. Оценивается и проверяется работа программ прототипа с целью приведения в соответствие с реальными запросами пользователей. Прототип проверяется на:

• удобство и адекватность интерфейсов ввода/вывода (характер вопросов в диалоге, связность выводимого текста результата и др.);

• эффективность стратегии управления (порядок перебора, использование нечеткого вывода и др.);

• качество проверочных примеров;

• корректность базы знаний (полнота и непротиворечивость правил).

• выявлений ошибок в подходе и реализации прототипа и выработка решений по доводке системы до промышленного варианта.

Средняя продолжительность 1-2 недели.

3.7.4. Развитие прототипа до промышленной ЭС

При неудовлетворительном функционировании прототипа эксперт и инженер по знаниям имеют возможность оценить, что именно будет включено в разработку окончательного варианта системы.

Если первоначально выбранные объекты или свойства оказываются неподходящими, их необходимо изменить. Можно сделать оценку общего числа эвристических правил, необходимых для создания окончательного варианта экспертной системы. Иногда [Хювенен, Сеппянен, 1991] при разработке промышленной и/ или коммерческой системы выделяют дополнительные этапы для перехода (табл. 3.2),

Демонстрационный прототип — Действующий прототип – Промышленная система —Коммерческая система.

Однако чаще реализуется плавный переход от демонстрационного прототипа к промышленной системе, при этом, если программной инструментарий был выбран удачно, не обязательно даже переписывать окончательный вариант другими программными средствами.

Понятие же коммерческой системы в нашей стране входит в понятие «промышленный программный продукт», или «промышленная ЭС» (в этой работе).

Основная работа на данном этапе заключается в существенном расширении базы знаний, то есть в добавлении большого числа дополнительных правил, фреймов, узлов семантической сети или других элементов знаний. Эти элементы знаний обычно увеличивают глубину системы, обеспечивая большее число правил для трудноуловимых аспектов отдельных случаев, В то же время эксперт и инженер по знаниям могут увеличить базу знаний системы, включая правила, управляющие дополнительными подзадачами или дополнительными аспектами экспертной задачи (метазнания).

Таблица 3.2

Переход от прототипа к промышленной экспертной системе

Система	Описание
Демонстрационный прототип ЭС	Система решает часть задач, демонстрируя жизнеспособность подхода (несколько десятков правил или понятий)
Исследовательский прототип ЭС	Система решает большинство задач, но неустойчива в работе и не полностью проверена (несколько сотен правил или понятий)
Действующий прототип ЭС	Система надежно решает все задачи на реальных примерах, но для сложной задачи требует много времени и памяти
Промышленная система	Система обеспечивает высокое качество решений при минимизации требуемого времени и памяти; переписывается с использованием более эффективных средств представления знаний

Коммерческая система	Промышленная система, пригодная к продаже, то есть хорошо документирована и снабжена сервисом

После установления основной структуры ЭС знаний инженер по знаниям приступает к разработке и адаптация интерфейсов, с помощью которых система будет общаться с пользователем и экспертом. Необходимо обратить особое внимание на языковые возможности интерфейсов, их простоту и удобство для управления работой ЭС. Система должна обеспечивать пользователю возможность легким и естественным образом уточнять непонятные моменты, приостанавливать работу и т. д. В частности, могут оказаться полезными графические представления.

На этом этапе разработки большинство экспертов узнают достаточно о вводе правил и могут сами вводить в систему новые правила. Таким образом, начинается процесс, во время которого инженер по знаниям передает право собственности и контроля за системой эксперту для уточнения, детальной разработки и обслуживания.

3.7.5. Оценка системы

В завершение этапа разработки промышленной экспертной системы необходимо провести ее тестирование в отношении критериев эффективности. К тестированию широко привлекаются другие эксперты с целью апробирования работоспособности системы на различных примерах. Экспертные системы оцениваются главным образом для того, чтобы проверить точность работы программы и ее полезность. Оценку можно проводить, исходя из различных критериев, которые сгруппируем следующим образом:

- критерии пользователей (понятность и «прозрачность» работы системы, удобство интерфейсов и др.);
- критерии приглашенных экспертов (оценка советов-решений предлагаемых системой, сравнение ее с собственными решениями, оценка подсистемы объяснений и др.);
- критерии коллектива разработчиков (эффективность реализации, производительность, время отклика, дизайн, широта охвата предметной области, непротиворечивость БЗ, количество тупиковых ситуаций, когда система не может принять решение, анализ чувствительности программы к незначительным изменениям в представлении знаний, весовых коэффициентах, применяемых в механизмах логического вывода, данных и т, и.).

3.7.6. Стыковка системы

На этом этапе осуществляется стыковка экспертной системы с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать. Иногда это означает внесение существенных изменений. Такие изменения требуют непременного вмешательства инженера по знаниям или какого-либо другого специалиста, который сможет модифицировать систему. Под стыковкой подразумевается также разработка связей между экспертной системой и средой, в которой она действует.

Когда экспертная система уже готова, инженер по знаниям должен убедиться в том, что эксперты, пользователи и персонал знают, как эксплуатировать и обслуживать ее. После передачи им своего опита в области информационной технологии инженер по знаниям может полностью предоставить ее в распоряжение пользователей.

Для подтверждения полезности системы важно предоставить каждому из пользователей возможность поставить перед ЭС реальные задачи, а затем проследить, как она выполняет эти задачи. Для чего чтобы система была одобрена, необходимо представить ее как помощника, освобождающего пользователей от обременительных задач, а не как средство их замещения.

Стыковка включает обеспечение связи ЭС с существующими базами данных и другими системами на предприятии, а также улучшение системных факторов$_t$ зависящих от времени$_t$ чтобы можно было обеспечить ее более эффективную работу и улучшить характеристики ее технических средств, если система работает в необычной среде (например, связь с измерительными устройствами).

Пример3.7

Успешно состыкована со своим окружением система PUFF - экспертная система для диагностики заболеваний легких [Хейес-Рот и др., 1967]. После того как PUFF была и все были удовлетворены ее работой, систему перекодировали с LISP на Бейсик. Затем систему перенесли на ПЭВМ, которая уже работала в больнице. В свою очередь, эта ПЭВМ была связана с измерительными приборами. Данные с измерительных приборов сразу поступают в ПЭВМ. PUFF обрабатывает эти данные и печатает рекомендации Для врача. Врач в принципе не взаимодействует с PUFF. Система полностью интегрирована со своим окружением - она представляет собой интеллектуальное расширение аппарата исследования легких, который врачи давно используют.

Пример 3.8

Другой системой, которая хорошо функционирует в своем окружении, является CAT-1 [Уотер мен, 1990] - экспертная система для диагностики неисправностей дизелей локомотивов.

Эта система была разработана также на JJSP, а затем была переведена на FORTH с тем, чтобы ее можно было более эффективно использовать в различных локомотивных целях. Мастер по ремонту запрашивает систему о возможных причинах неисправности дизеля. Система связана с видеодиском, с помощью которого мастеру показывают визуальные объяснения и подсказки, касающиеся более подробных проверок, которые он должен сделать.

Кроме того, если оператор не уверен в том, как устранить неисправность, система предоставляет ему обучающие материалы, которые фирма подготовила предварительно, и показывает ему их на видеотерминале. Таким образом, мастер по ремонту может с помощью экспертной системы диагностировать проблему, найти тестовую процедуру, которую он должен использовать, получить на дисплее объяснение, как провести тест, или получить инструкции о том, как справиться с возникшей проблемой.

3.7.7. Поддержка системы

При перекодировании системы на язык, подобный С, повышается ее быстродействие и увеличивается переносимость, однако гибкость при этом уменьшается. Это приемлемо лишь в том случае, если система сохраняет вес знания проблемной области и это знание не будет изменяться в ближайшем будущем. Однако если экспертная система создана именно из-за того, что проблемная область изменяется то необходимо поддерживать систему в ее инструментальной среде разработки.

Пример 3.9

Удачным примером ЭС, внедренной таким образом, является *XCQK* (Rl) - 3С,которую фирма DEC использует для комплектации ЭВМ семейства VAX. Одной из ключевых проблем, с которой столкнулась фирма DEC, является необходимость постоянного внесения изменений для новых версий оборудования, новых спецификаций и т. д. Для этой цели XCON поддерживается в программной среде OPS5.

Вопросы для повторения

1. Как называются сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных области и тиражирующие этот эмпирический опыт для консультаций менее квалифицированные пользователей?
2. Опишите обобщенную структуру экспертных систем и назовите ее обязательные компоненты.
3. Как можно представить процесс функционирования экспертных систем?
4. Перечислите основные термины в области разработки экспертных систем и дайте им определение.
5. Что собой представляет и для чего нужен интерфейс пользователя?
6. Как называется ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному)?
7. Для чего нужен решатель, и как еще его называют?
8. Какое назначение имеет система объяснений? Ответы на какие вопросы пользователя позволяет получить система объяснений?
9. Какие возможности предоставляет инженеру по знаниям интеллектуальный редактор базы знаний? Какие компоненты он в себя включает?
10. По каким признакам осуществляется классификация экспертных систем?
11. Перечислите типы задач, решение которых поддерживается с помощью экспертных систем?
12. Какие виды экспертных систем выделяются в классификации по связи с реальным временем?
13. Как называются экспертные системы, интерпретирующие ситуацию, которая меняется с некоторым фиксированным интервалом времени?
14. Как работают динамические экспертные системы?
15. Как классифицируются экспертные системы в зависимости от степени интеграции с другими программами?
16. Чем отличаются автономные и гибридные экспертные системы?
17. Как называют группу специалистов, ответственных за создание экспертной системы?
18. Кто входит в коллектив разработчиков экспертных систем?
19. Какие условия должны соблюдаться для достижения эффективного сотрудничества в коллективе разработки экспертных систем?
20. Какими психофизическими и профессиональными характеристиками должен обладать пользователь, как член коллектива разработки экспертных систем?
21. Какими желательными качествами должны обладать эксперт и программист, как члены коллектива разработки экспертных систем?
22. Какова роль инженера по знаниям в коллективе разработки экспертных систем?
23. Перечислите формы знаний, с которыми работает инженер по знаниям.
24. Зависит ли эффективность и продолжительность процесса разработки экспертной системы от успешности выбора и подготовки коллектива разработки?
25. Назовите основные этапы разработки экспертной системы и дайте им характеристику.
26. Перечислите стадии разработки прототипа. Какое назначение имеет прототипная система?
27. Как осуществляется переход от прототипа к промышленной экспертной системе?

По каким критериям происходит оценка экспертной системы?

Резюме по теме

Представление знаний и разработка систем, основанных на знаниях (knowledge-based systems) - это основное направление в области изучения искусственного интеллекта. Оно связано с разработкой моделей представления знаний, созданием баз знаний, образующих ядро экспертных систем. В последнее время включает в себя модели и методы извлечения и структурирования знаний и сливается с инженерией знаний. Центральная парадигма интеллектуальных технологий сегодня — это обработка, знаний. Системы, ядром которых является база знаний или модель предметной области, описанная на языке сверхвысокого уровня, приближенном к естественному, называют интеллектуальными. Наиболее распространенным видом интеллектуальных систем являются экспертные системы.

Глоссарий

CASE-технологии	автоматизированный инжиниринг программных средств, то есть совокупность методик проектирования и сопровождения программных средств на всем из жизненном цикле, поддержанная взаимоувязанными средствами автоматизации.
ER-моделирование	метод разработки ER-модели, отражающей информационные потребности предприятия-заказчика и, не зависящей от физической реализации хранилищ данных и метода доступа к ним
Агрегат	именованная совокупность элементов или других агрегатов
Анализ	в течение которого разрабатываются детальные модели предметной области, формируются точные требования к будущей программной системе и закладывается точная основа для перехода к проектированию
База данных	совокупность взаимосвязанных данных, используемых группой пользователей и хранящаяся с регулируемой избыточностью.
База знаний (БЗ)	ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному)
Базовый запрос	запрос, который является источником данных для другого запроса, формы, отчета или страницы доступа к данным.
Базовая таблица	таблица, являющаяся источником данных запроса, формы, отчета или страницы доступа к данным.
Гиперссылка	текст, выделенный цветом или подчеркиванием, или графическое изображение, при щелчке на котором осуществляется переход к файлу, определенному месту в файле, странице HTML в World Wide Web или странице HTML во внутренней сети (интранет).
Группировка	разделение данных на группы, по определенному критерию.
Данные	произвольная информация, представленная в символьной (цифровой) форме.

Запись	агрегат, который не входит в состав никакого другого и составляет основную единицу обработки БД, то есть записи обновляются, извлекаются и удаляются
Запрос	обращение к СУБД, содержащее задание на выборку, добавление, изменение или удаление записей.
Знания	это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.
Индекс	средство, обеспечивающее быстрый доступ к данным в таблице на основе значений одного или нескольких столбцов. Индекс представляет собой упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения.
Инженер по знаниям	специалист в области искусственного интеллекта, выступающий в роли промежуточного буфера менаду экспертом и базой знаний. Синонимы: *криптолог, инженер-интерпретатор, аналитик*
Интеллектуальный редактор БЗ	*Интеллектуальный редактор БЗ* — программа, представляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок («help» — режим) и других сервисных средств, облегчающих работу с базой.
Интерфейс пользователя	комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.
Инфологический уровень	второй уровень абстракции, представляющий собой интеграцию ЛПП, соответствующую взгляду на предметную область ее администратора**.**
Искусственный интеллект	одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными задачи, общаясь с ЭВМ на ограниченном подмножестве естественного языка.
Каскадное обновление	средство поддержания целостности данных в связанных таблицах, которое при изменении значения ключевого поля в главной таблице обеспечивает обновление всех связанных записей в подчиненной таблице.
Ключевое поле	поле, которое однозначно идентифицирует каждый объект в таблице, т. е. позволяет четко отличить один объект от другого. Может быть частью составного ключа.
Кнопочная форма	средство, обеспечивающее пользователям доступ к функциям приложения.
Концептуальный	соответствует представлению о логической организации данных администратора БД. На данном уровне существует привязка к модели данных и средствам ее реализации.

уровень	
Ленточная форма	форма, в которой выводится одновременно несколько записей из базовой таблицы или запроса.
Лингвистическая переменная (ЛП)	переменная, значение которой определяется набором вербальных (то есть словесных) характеристик некоторого свойства.
Линейный список	предназначен для создания сложных динамических структур, организованных с помощью ссылок в определенном порядке.
Локальные пользовательские представления (ЛПП)	начальный уровень абстракции соответствует представлениям о предметной области конечных пользователей
Макрос	структура, состоящая из одной или нескольких макрокоманд, которые выполняются либо последовательно, либо в порядке, заданном определенными условиями.
Модель данных	совокупность принципов организации базы данных. В СУБД Access используется реляционная модель данных.
Модуль	объект базы данных Access, содержащий программный код на языке VBA: декларации переменных и функций.
Нормализация отношений	это пошаговый обратимый процесс разложения исходных отношений БД на более мелкие и простые отношения. Каждая нормальная форма (НФ) ограничивает типы допустимых функциональных зависимостей отношений
Отчет	объект базы данных, который используется для вывода на печать данных в отформатированном виде.
Первичный ключ	один или несколько столбцов (атрибутов), которые однозначно идентифицируют каждую запись в таблице, т. е. позволяют четко отличить одну запись от другой.
Пользователь	специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС
Подсхема	описание данных на внешнем уровне представления
Поле	элемент данных в записи.
Предметная область	это часть реального мира, подлежащая изучению с целью организации управления и в конечном счете автоматизации.

Проектирование	стадия точного плана реализации требований, определенных ранее
Прототипная система	является усеченной версией экспертной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта
Реализация	стадия, когда выполняется непосредственная реализация и сборка программной системы
Реляционная СУБД	СУБД, базирующаяся на реляционной модели данных, в которой связи между наборами данных реализованы на основании совпадения значений полей.
Решатель	программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: *дедуктивная машина, машина вывода, блок логического вывода,*
Связь	логическое отношение между объектами, представленными таблицами.
Сервер	это компьютер, устройство, ресурс, который может отдавать все свои ресурсы или часть другим пользователям.
***Система* объяснений**	программа, позволяющая пользователю получить ответ на вопросы: «Как была получена та или иная рекомендация?» и «Почему и прияла такое решение?»
Система управления базами данных (СУБД)	комплекс программ и языковых средств, предназначенных для создания, ведения и использования баз данных.
Сортировка	изменение порядка, в котором представлены данные.
Сопровождение	состоит в том, что ИС поддерживается в актуальном состоянии, если в связи с извлечением в предметной области потребуются значительные доработки, весь жизненный цикл повторяется сначала
Стек	динамическая структура данных, основанная на принципе LIFO (Last In – First Out). По аналогии с магазином автомата стек так же называют стек
Страница	участок памяти для хранения подмножества данных.
Стратегическое планирование	за короткое время изучается деловая деятельность заказчика и формируется общее представление о ней
Схема данных	совмещение в СУБД описания логических и физических уровней
Таблица базы данных	набор данных в реляционной СУБД. Состоит из переменного количества записей постоянной структуры

Тип данных	атрибут поля в таблице, который определяет, какие данные могут содержаться в этом поле.
Файл	логическое понятие, относящееся в основном к организации данных, его употребляют, когда конкретный физический носитель либо не интересен, либо всегда одинаков
Фильтрация данных	позволяет показать или скрыть записи удовлетворяющие, неудовлетворяющие определенному условию
Формы	средства ввода и отображения данных, содержащие средства управления, помощью которых осуществляется доступ к данным.
Целостность данных	логическая непротиворечивость данных одной и другой таблицы
Хранимая запись	состоит из служебной и информационной частей
Экспертные системы (ЭС)	сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных области и тиражирующие этот эмпирический опыт для консультаций менее квалифицированные пользователей
Элемент данных	наименьшая единица структуры данных

Оглавление